# MATLAB 7.0 实用指南
## （上册）

苏金明　王永利　编著

電子工業出版社
**Publishing House of Electronics Industry**
北京 • BEIJING

## 内 容 简 介


本套书基于 MATLAB 的最新版本 7.0 分上、下两册详细介绍该软件的使用方法，主要内容包括 MATLAB 7.0 的入门知识、界面设计、编译、接口，以及新版本变化较大的图形功能和图像处理、虚拟现实、地图制作等 3 个工具箱。

本书为上册，主要介绍 MATLAB 7.0 的工作环境、数组、矩阵、数值计算、M 文件、图形用户界面设计、编译、接口及二维、三维图形绘制功能，以及最新的编译器 4.0、COM 生成器 1.1 和图形的交互创建及编辑功能；从图形系统开发的角度介绍二维图形定制和三维图形建模、着色、光照、材质、纹理和交互处理。此外，还介绍了一些实用的科学计算可视化工具。

本书内容全面，新颖，适合大学生、研究生、科研人员和科技工作者阅读参考。





**图书在版编目（CIP）数据**

MATLAB 7.0 实用指南. 上册/苏金明，王永利编著. —北京：电子工业出版社，2004.10
ISBN 7-121-00433-X

Ⅰ. M… Ⅱ. ①苏… ②王… Ⅲ. 计算机辅助计算—软件包，MATLAB 7.0 Ⅳ. TP391.75

中国版本图书馆 CIP 数据核字（2004）第 102643 号

责任编辑：龚兰方
印　　刷：北京天宇星印刷厂
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路 173 信箱　邮编　100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：21.75　字数：557 千字
印　　次：2004 年 11 月第 1 次印刷
印　　数：5000 册　　定价：33.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

# 前　言

近年来，MATLAB 以其强大的矩阵计算和图形可视化功能逐渐为国人所知。很多学校已经开设这方面的课程，很多学生已经开始使用该软件完成论文设计。科学计算软件的使用，极大地提高科研人员的工作效率，可以更快、更准确地完成计算方案的设计，可以在必要的时候用图形图像表示计算结果和描述运行机制。

本书基于 MATLAB 7.0 版本，分上、下两册介绍该软件的使用。相对于以前诸版本，7.0 版本在图形和编译器方面有比较明显的变化，部分工具箱也有一些变化。本书的主要内容可以概括为两个部分，一部分系统介绍 MATLAB 的基础和核心功能，即 MATLAB 总包的功能；另一部分系统介绍新版本变化最大的图形图像功能。上册主要介绍 MATLAB 总包的应用，下册主要介绍几个图形图像方面的工具箱。

第 1～第 16 章为上册的主要内容，系统地介绍 MATLAB 7.0 的基本特点、运行环境、数组、矩阵、数值计算、M 文件、图形用户界面设计、编译、接口，以及二维、三维图形功能。第 1～第 5 章为比较基础的内容，适合于初学者入门；第 6 章介绍最新的编译器 4.0 和 COM 生成器 1.1。编译器 4.0 可以接受对象数据类型，这在以前是不行的。利用 COM 生成器可以将 MATLAB 的 M 文件和 MEX 文件打包成 COM 组件，这些组件又可以用于支持 COM 机制的应用程序，如 VC, VB 等，从而可以实现无缝集成。

MATLAB 7.0 的最大亮点就在于添加了图形的交互创建和编辑功能。这里所说的交互，指的是鼠标交互，即主要通过鼠标的单击和拖拉操作完成图形的绘制和编辑。交互功能的添加，提高了绘图效率和绘图准确性。与此相对应，作为 MATLAB 图形图像和界面基础的句柄图形对象也有了很大的改变。这种改变，体现在对象抽象和对象组织上。

具体来说，图形部分的内容包括二维图形绘制、图形的交互创建和编辑、二维图形的定制、三维模型和场景的创建和变换，以及 MATLAB 提供的一系列科学计算可视化工具等。二维部分，MATLAB 可以绘制条形图、等值线图、向量图等几十种图形，利用图形对象创建函数，还可以实现图形定制；三维部分，可以创建三维网格图、曲面图、流线图、剖面图、等值面图等多种图形。三维程序的开发，是一件富有挑战性但又其乐无穷的事情。在这方面，MATLAB 实际上提供了一个比较高的平台。本书分表面模型和多边形模型两种情况，全面介绍三维模型的创建、着色、光照、材质、透明性、纹理映射和交互操作。

第 17～第 42 章为下册的主要内容，主要介绍 MATLAB 的图像处理、虚拟现实和地图制作工具箱。

第 17～第 30 章介绍图像处理工具箱，内容包括图像合成、空间变换、邻域和块处理、线性滤波和滤波器设计、基于区域的处理、变换域处理、数学形态学、图像分析、图像增强、图像配准和图像恢复等。

第 31～第 35 章介绍虚拟现实工具箱，内容包括虚拟场景的创建、浏览和交互。

第 36～第 42 章介绍地图制作工具箱，例如，地理空间数据、地理空间几何和地图投影等基础知识和实现方法，还介绍如何利用地图制作工具箱绘制和定制二维、三维地图。

目前，虚拟现实在科研方面迅速地向很多专业领域渗透，是当前计算机图形学研究的三大热点之一。传统的实现方法是使用 OpenGL, DirectX3D 等 API 和 VRML,VEGA 等语言，需要使用者具有较多的知识储备。而 MATLAB 的虚拟现实工具箱提供专门的 VRML 编辑器和虚拟场景查看器，可以在不懂 VRML 语言的情况下实现虚拟场景的创建和浏览，并且这个虚拟场景还可以与 MATLAB 交互，因而是可控的。对于广大专业工程技术研究人员来说，这无疑是一个福音。

在“数字化地球”的时代，电子地图的制作是热点。MATLAB 的地图制作工具箱提供了制作电子地图的一种途径。该工具箱的功能强大，可以绘制三维地图。

在编写过程中，作者力求全书思路清晰，结构合理，叙述流畅，术语地道，实例丰富，并诚挚地希望能收到抛砖引玉的效果。如果你看了书以后很有想法，我们可以交流；如果很有收获，甚至做出一个很好的三维系统，我们愿意分享你的快乐！

本书适合于对 MATLAB 感兴趣的大学生、研究生、教师和科研技术人员阅读。

写作过程中得到了很多读者朋友和网友的热心支持，表示感谢！另外，还要感谢黄国明、刘波、王卫、刘玉珊等给予的帮助！

由于水平有限，书中缺点和错误之处在所难免，谨请读者朋友批评指正！可通过电子邮件与我们联系：

苏金明 s_jm@263.net.cn

王永利 wangyl@cdut.edu.cn

编 著 者

# 目　录

# 第 1 章 MATLAB 7.0 简介

MATLAB 是由 MathWorks 公司于 1984 年推出的一套科学计算软件，分为总包和若干个工具箱。它具有强大的矩阵计算和数据可视化能力，一方面可以实现数值分析、优化、统计、偏微分方程数值解、自动控制、信号处理等若干个领域的数学计算，另一方面可以实现二维、三维图形绘制，三维场景创建和渲染、科学计算可视化、图像处理、虚拟现实和地图制作等图形图像方面的处理。MATLAB 的最新版本是 7.0。

## 1.1 MATLAB 的特点

下面概略介绍 MATLAB 的基本特点和 MATLAB 7.0 的新特点，以便对 MATLAB 及其最新版本有一个总体认识。

### 1.1.1 MATLAB 的基本特点

近年来，MATLAB 在国内的知名度越来越大，并已被广泛地应用于教学和科研。该软件的特点可以归纳为以下几点。

（1）简单易学　MATLAB 是一门编程语言，其语法规则与一般的结构化高级编程语言，如 C 语言等大同小异，而且使用更方便，具有一般语言基础的用户很快就可以掌握。

（2）代码短小高效　由于 MATLAB 已经将数学问题的具体算法编成了现成的函数，用户只要熟悉算法的特点、使用场合、函数的调用格式和参数意义等，通过调用函数很快就可以解决问题，而不必花大量的时间纠缠于具体算法的实现。

（3）计算功能非常强大　该软件具有强大的矩阵计算功能，利用一般的符号和函数就可以对矩阵进行加、减、乘、除运算以及转置和求逆运算，而且可以处理稀疏矩阵等特殊的矩阵，非常适合于有限元等大型数值算法的编程。此外，该软件现有的数十个工具箱，可以解决应用中的大多数数学问题。

（4）强大的图形表达功能　该软件不仅可以绘制一般的二维三维图形，如线图、条形图、饼图、散点图、直方图、误差条图等，还可以绘制工程特性较强的特殊图形，如玫瑰花图、极坐标图等。科学计算要涉及到大量的数据处理，利用图形展示数据场的特征，能显著提高数据处理的效率，提高对数据反馈信息的处理速度和能力。MATLAB 提供了丰富的科学计算可视化功能，利用它，可以绘制二维三维矢量图、等值线图、三维表面图、假彩色图、曲面图、云图、二维三维流线图、三维流锥图、流沙图、流带图、流管图、卷曲图、切片图等，此外还可以生成快照图和进行动画制作。基于 MATLAB 句柄图形对象，结合绘图工具函数，可以根据需要用 MATLAB 绘制自己的图形。

（5）可扩展性能　可扩展性能是该软件的一大优点，用户可以自己编写 M 文件，组成自己的工具箱，方便地解决本领域内常见的计算问题。此外，利用 MATLAB 编译器和运行

时服务器，可以生成独立的可执行程序，从而可以隐藏算法并避免依赖 MATLAB。MATLAB 支持 DDE 和 ActiveX 自动化等机制，可以与同样支持该技术的应用程序接口。

### 1.1.2 MATLAB 7.0 的新特点

MATLAB 是一种高级计算语言，是进行数据分析和算法与应用开发的交互式开发环境。MATLAB 7.0 在编程、代码效率、图形、计算、数据获取和运行等方面主要有下面几个新特点。

**1. 开发环境**

- 提供了新的桌面。新桌面提供了多文档管理、锚定图形窗口以及保存定制输出和常用命令快捷键的能力。
- 改进了数组编辑器和工作空间浏览器，使得察看、编辑变量和用变量数据绘图更加容易。
- 可以在编辑器中执行一部分 M 代码。
- 自动将 M 代码发布为 HTML, Word 或 LaTex 文档。

**2. 编程**

- 可以创建嵌套函数，它提供了定义和调用自定义函数的一种更便捷的途径。
- 提供了在命令行或脚本式 M 文件中定义单行函数的隐函数表示形式。
- 提供了用标准调用语法，而不是 feval 调用函数句柄的能力。
- 使用条件断点，可以在条件表达式为真时停止运行。

**3. 计算**

- 整数计算部分，可以在计算和处理更大的整型数据集时保持数据类型。
- 单精度计算、FFT 和滤波这几部分，可以处理更大的单精度数据集。
- 计算几何部分，可以使用更稳健的函数，它对算法选择给出了更多控制。
- 使用 linsolve 函数，通过指定矩阵系数的结构，可以更快地求解线性方程组。
- ODE 求解器可以控制隐式差分方程和多点边界值问题。

**4. 图形**

- 使用新的绘图界面，可以在不输入 M 代码的情况下交互式地创建和编辑图形。
- 可以自动生成图形的 M 代码。这样，可以利用该代码重建图形。
- 图形标注作了改进，包括绘制图形、对象对齐和将标注“钉”到数据点。
- 提供了数据探察工具，包括图形平移和数据光标等。
- 可以对成组的图形对象进行旋转、平移和缩放等变换。
- 可以从 GUIDE 获取用户界面面板和 ActiveX 控件。

**5. 数据获取和外部接口**

- 提供了读取很大的文本文件和写为 Excel 和 HDF5 文件的文件输入输出函数。
- 提供了压缩 MAT 文件的选项，使得可以用更少的磁盘空间保存大的数据。
- 使用 javaaddpath 函数可以在不重新启动 MATLAB 的情况下动态添加、删除和重新载入 Java 类。
- COM 定制接口、服务器事件和 Visual Basic 脚本支持。
- 可以基于 SOAP 获取 Web 服务。
- 提供了可以连接到 FTP 服务器进行远程文件操作的 FTP 对象。

● MAT 文件中的字符数据可以用于多种语言。

## 1.2 MATLAB 桌面简介

启动 MATLAB 时，第一件事就是查看 MATLAB 桌面。它由几个管理文件、变量和与 MATLAB 相关的应用程序的工具组成。

第一次启动 MATLAB 时，桌面如图 1-1 所示。可以根据需要改变桌面外观设置，包括移动、缩放和关闭工具窗口。

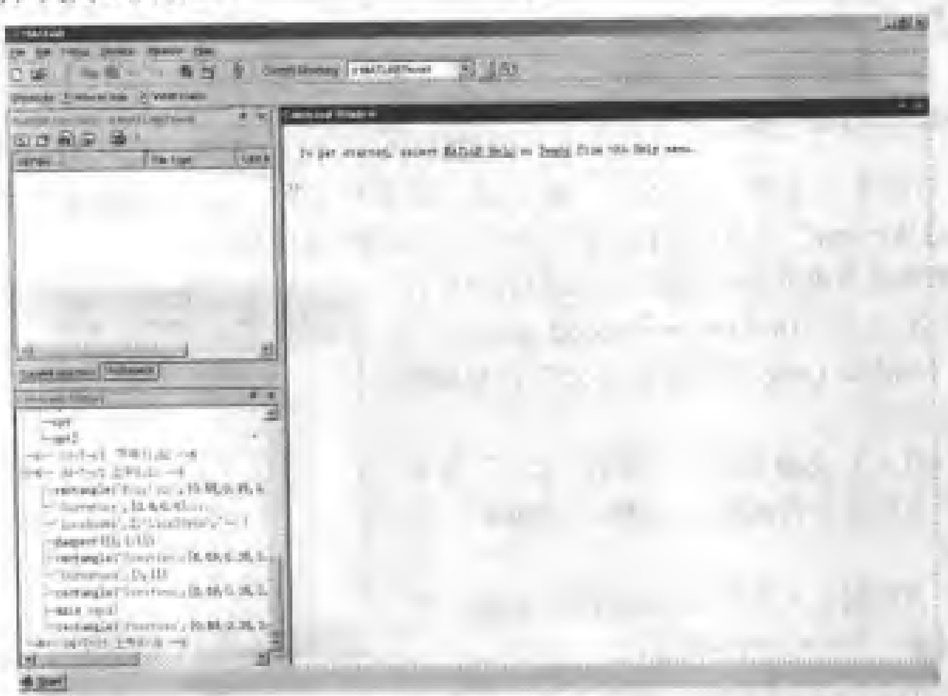


图 1-1 MATLAB 桌面

MATLAB 桌面包括表 1-1 中的几种工具窗口，在默认情况下，它们中间有一些没有显示。如果更喜欢命令行界面，可以用等价的函数完成桌面工具可以完成的任务。

表 1-1 MATLAB 的桌面工具

| 桌面工具 | 描述 |
|---|---|
| 数组编辑器 | 查看表格形式的数组内容并编辑数组的值 |
| 命令窗口 | 运行 MATLAB 函数 |
| 命令历史窗口 | 显示命令窗口中键入的命令，可以从该窗口中复制和运行命令 |
| 当前路径浏览器 | 查看文件，进行打开、查找和管理等操作 |
| 编辑器/调试器 | 创建、编辑和调试 M 文件(包含 MATLAB 函数的文件) |
| 图形窗口 | 创建、修改、查看和打印 MATLAB 图形 |
| 帮助浏览器 | 查看和搜索所有 MathWorks 产品的文档 |
| Profiler 窗口 | 用图形界面改进 M 文件的运行 |
| 启动按钮 | 运行工具和获取所有 MathWorks 产品的文档，并创建和使用 MATLAB 快捷方式 |
| Web 浏览器 | 查看 HTML 和与 MATLAB 相关的信息 |
| 工作空间浏览器 | 查看和改变工作空间中的内容 |

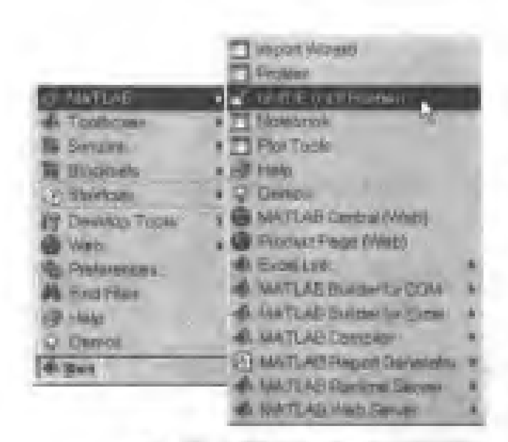


图 1-2　“Start” 菜单

### 1.2.1　启动按钮

打开 MATLAB 主界面以后，单击“Start”按钮或者说启动按钮，显示一个菜单，如图 1-2 所示。也可以通过同时按下<Alt>键和<S>键来察看“Start”菜单的内容。

利用“Start”菜单及其子菜单中的选项，可以直接打开 MATLAB 的相关工具。

### 1.2.2　命令窗口

命令窗口是用于输入数据，运行 MATLAB 函数和脚本并显示结果的主要工具之一。命令窗口没有打开时，从“Desktop”菜单中选择“Command Window”选项可以打开它。命令窗口如图 1-3 所示。




图 1-3　命令窗口

如果更喜欢简单的没有其他工具窗口的命令行界面，依次选择 Desktop→Desktop Layout→Command Window Only 菜单选项。“>>”符号是输入函数的提示符。

当 MATLAB 在命令窗口中显示“K>>”提示符时，表示当前处于调试模式。键入“dbquit”，返回正常模式。

在提示符后面输入数据和运行函数。例如，下面的代码创建一个 3×3 的矩阵 *A*。

```
A = [1 2 3; 4 5 6; 7 8 10]
```

输入命令行以后单击回车键，MATLAB 返回矩阵 A 的值：

```
A =
     1     2     3
     4     5     6
     7     8    10
```

运行函数时，键入函数及其变量然后单击回车键。例如，在命令行中输入

```
magic(2)
```

MATLAB 返回

```
ans =
     1     3
     4     2
```

在操作系统命令前添加“!”，可以在不退出 MATLAB 的情况下调用工具或其他可执行程序。例如，

```
!dir
```

可以调用 DOS 系统的“dir”命令。如果希望外部程序完成任务或退出以后返回

MATLAB，在命令行末尾添加“&”字符，例如，

```
!dir &
```

将输出显示在单独的窗口中或者以背景模式运行应用程序。又如，

```
!excel.exe &
```

打开 Excel 并返回 MATLAB 命令窗口，可以继续运行 MATLAB 的语句。

### 1.2.3　命令历史窗口

命令历史窗口显示命令窗口中最近输入的所有语句。命令历史窗口如图 1-4 所示。

注明每次使用时的时间

单击-，隐藏所属选项；单击+，显示所属选项

选择一行或多行并右击鼠标，可以进行复制等操作

图 1-4　命令历史窗口

如果当前命令历史窗口没有显示，用“Desktop”菜单打开它。也可以用 commandhistory 命令打开。

可以将命令历史窗口中的语句复制到命令窗口或其他窗口中。直接双击命令历史窗口中的语句，可以执行该语句。

### 1.2.4　工作空间窗口

MATLAB 工作空间由一系列变量组成，如图 1-5 所示。

可以通过使用函数、运行 M 文件和载入已经存在的工作空间来添加变量。例如，输入

```
t = 0:pi/4:2*pi;
y = sin(t);
```

工作空间会包括这两个变量 y 和 t，每个变量有 9 个值。

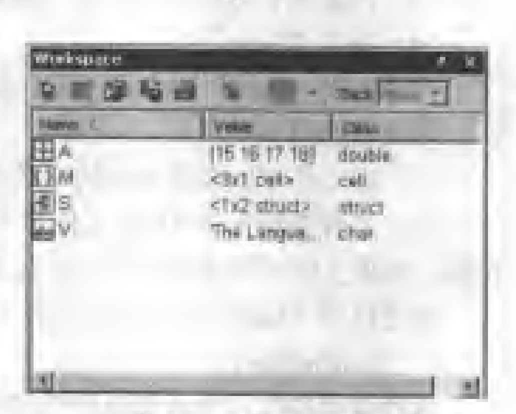



图 1-5　工作空间窗口

可以用工作空间浏览器显示每个变量的名称、值、数组大小、字节大小和类型。用 who 函数可以列出当前工作空间中的所有变量，用 whos 函数列出变量和它们的大小和类型等信息。例如，

```
who
    Your variables are:
```

```
    A M S V
whos
  Name        Size        Bytes   Class
   A          1x4         32      double array
   M          3x1         202     cell array
   S          1x2         598     struct array
   V          1x35        70      char array
Grand total is 76 elements using 902 bytes
```

用 exist 函数查看指定变量是否在工作空间中。

退出 MATLAB 时，工作空间中的内容随之清除。可以将当前工作空间中的部分或全部变量保存到一个 MAT 文件中，它是一种二进制文件，扩展名为 .mat。可以在以后使用它时载入它。

按照下面的步骤保存所有变量：

（1）从“File”菜单中选择“Save Workspace As”，或单击工作空间浏览器工具条中的按钮，打开“Save”对话框；

（2）指定保存路径和文件名。MATLAB 会自动提供 .mat 扩展名；

（3）单击“Save”按钮。

也可以通过从“File”菜单中选择“Save Workspace As”选项从桌面上保存工作空间变量。

按照下面的步骤保存当前工作空间中的部分变量：

（1）在工作空间浏览器中选择变量，按住<Shift>键的同时连续单击变量名，可以选择多个变量；

（2）右击并从弹出的上下文菜单中选择“Save As”选项，打开“Save to MAT-File”对话框；

（3）指定保存路径和文件名。MATLAB 自动提供.mat 扩展名；

（4）单击“Save”按钮。

也可以用“Save”命令保存工作空间。

按照下面的步骤载入工作空间：

（1）在工作空间浏览器中的工具条上单击按钮，打开“Open”对话框；

（2）选择要载入的 MAT 文件并单击“Open”按钮，文件中的变量全部载入到当前工作空间。如果文件中的变量名与已有变量名相同，则覆盖已有变量。

也可以用“load”命令打开已有工作空间，例如

```
load('matfile')
```

选择变量以后，单击按钮可以从工作空间中删除。单击按钮并选择图形类型，可以创建图形。

用数组编辑器可以以电子表格的形式查看和编辑一维或二维数值数组、字符串、字符串单元数组和结构。

在工作空间浏览器中选择要打开的变量，然后单击按钮，打开数组编辑器，如图 1-6 所示。




图 1-6 数组编辑器

现在可以像在 Excel 中那样实现数据的复制、剪切、粘贴等操作。总的来讲，数组编辑器的优越性在于提供了一种可视的数据编辑方式。

### 1.2.5 当前目录浏览器

用 MATLAB 当前目录浏览器搜索、察看、打开、查找和改变 MATLAB 相关的路径和文件。在 MATLAB 桌面上，从"Desktop"菜单中选择"Current Directory"选项，或者在命令窗口键入"filebrowser"，打开当前目录浏览器，如图 1-7 所示。




图 1-7 当前目录浏览器

使用当前目录浏览器可以完成下面的主要任务：

- 察看和改变路径；
- 创建、重命名、复制和删除路径和文件；
- 打开、运行和察看文件的内容；

- 查找文件和文件中的内容；
- 获取源控制特性。

## 1.3　MATLAB 的帮助系统

完善的帮助系统是一个成熟的软件必不可少的内容，它有助于用户自学、进行在线查询和答疑解惑。

### 1.3.1　帮助浏览器

使用帮助浏览器可以搜索和察看所有 MathWorks 产品的文档和 demo。帮助浏览器是集成到 MATLAB 桌面的一个 HTML 察看器。在桌面工具条上单击?按钮，可以打开帮助浏览器，如图 1-8 所示。

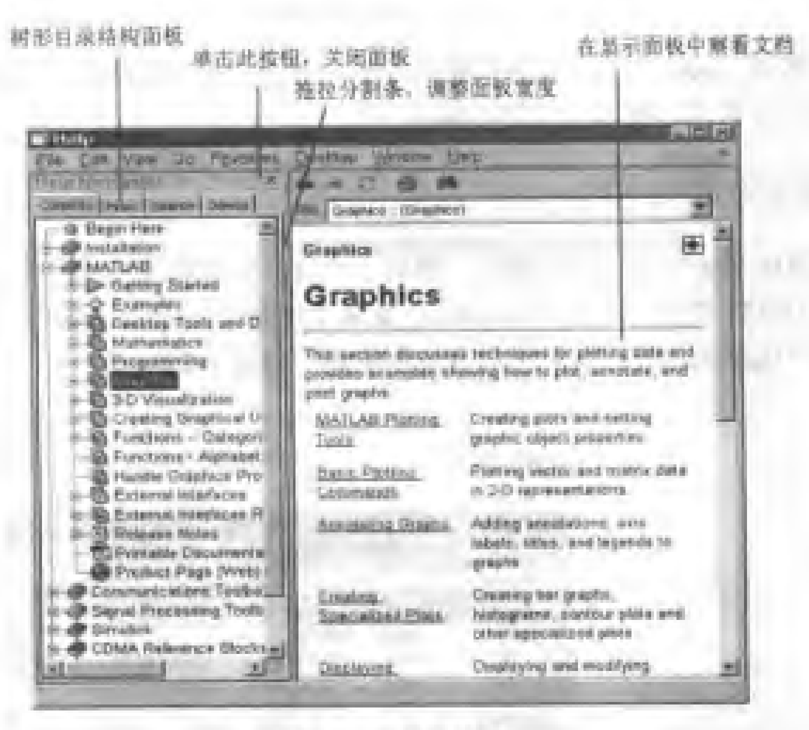



图 1-8　帮助浏览器

帮助浏览器主要由左右两个面板组成，一个是树形目录结构面板，用于查找信息；另一个是显示面板，在这里显示和察看信息。树形目录结构面板有 4 个选项卡，介绍如下：

- Contents 选项卡　察看文档内容的标题和目录；
- Index 选项卡　根据指定的关键词在文档中进行查找；
- Search 选项卡　在文档中查找指定的单词；
- Demos 选项卡　察看和运行 MathWorks 产品的演示程序。

### 1.3.2　help 函数和 doc 函数

除了帮助浏览器外，还可以使用 help 函数获取帮助。在命令窗口键入 help 命令，可以

获得当前安装的 Mathworks 产品的简单信息，在 help 后面跟工具箱或函数的名称，可以进一步获取更详细的帮助，帮助内容显示在命令窗口中，如图 1-9 所示。

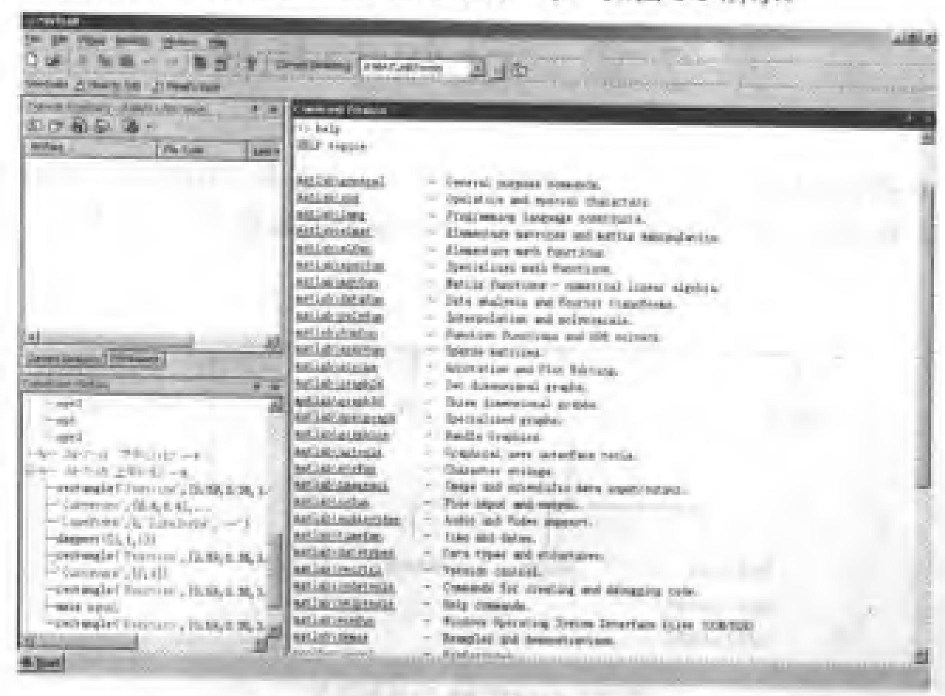


图 1-9　使用 help 命令

获取指定函数的帮助，可以使用 doc 函数。例如，doc format 语句在帮助浏览器中显示 format 函数的文档。

# 第 2 章　数组和矩阵

MATLAB 提供的数据类型如图 2-1 中所示，有十余种之多。但所有的 MATLAB 变量，不管它是什么类型的，都以数组或矩阵的形式保存。矩阵是数组的二维版本。

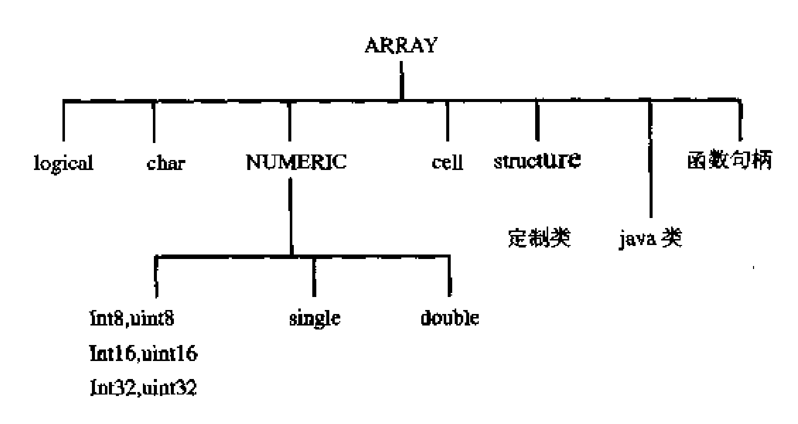



图 2-1　MATLAB 的数据类型

## 2.1　表达式

与其他程序语言类似，MATLAB 提供了数学表达式功能。但是，与大多数程序语言不同的是，这些表达式主要针对矩阵进行操作。与表达式相关的内容主要包括变量、数值表示、运算符和函数等。

### 2.1.1　变量

MATLAB 不需要任何类型声明和维数说明。对于新变量名，MATLAB 会自动创建对应的变量并分配合适的内存。如果变量已经存在，MATLAB 会改变它的内容；如果必要，会分配新的存储量。例如，

```
num_students = 25
```

创建一个 1×1 的名为 num_students 的矩阵，并将 25 作为元素的值。变量名的第一个字符必须是字母，后面可以跟个数不限的字符、数字或下划线。MATLAB 只使用变量名的前 31 个字符。MATLAB 区分大小写，如 A 和 a 是不同的变量。要察看赋给任何变量的矩阵，只需要简单地输入变量名就可以了。

### 2.1.2　数值表示

MATLAB 使用传统的数值表示方法。对于比较长的数，使用科学计数法，用字母 e 指定以 10 为底的幂次。虚数用 i 或 j 作为后缀。下面是一些合法的数值表示。

```
3            -99            0.0001
9.6397238    1.60210e-20    6.02252e23
1i           -3.14159j      3e5i
```

所有数值在内部保存为 IEEE 浮点标准指定的 long 型格式。浮点数据的精度为 16 个小数位，范围大约为 $10^{-308}$～$10^{+308}$。

### 2.1.3　运算符

表达式采用的算术运算符和优先规则如表 2-1 所示。

表 2-1　表达式采用的算术运算符和优先规则

| 运 算 符 | 说　明 |
|---|---|
| + | 加 |
| - | 减 |
| * | 乘 |
| / | 除 |
| \ | 左除 |
| ^ | 幂 |
| ' | 复数共轭转置 |
| () | 指定计算顺序 |

### 2.1.4　函数

MATLAB 提供了很多内部数学函数，包括 abs,sqrt,exp 和 sin 等。对负数取平方根或对数不会导致错误，MATLAB 会自动返回复数计算结果。MATLAB 还提供了很多高级的数学函数，包括 Bessel 和 gamma 函数等。这些函数中的大部分都接受复数变量。在命令窗口键入下面的命令，可以查看此类函数的列表。

```
help elfun
```

键入下面的命令行，可以找到更多的高级数学函数和矩阵函数。

```
help specfun
help elmat
```

有些函数如 sqrt 和 sin 是内部函数，内部函数是 MATLAB 内核的一部分，所以它们的计算效率很高，但计算细节无法获取。其他函数，如 gamma 和 sinh，是用 M 文件实现的。内部函数和其他函数有一些不同，例如，对于内部函数，无法看到代码；对于其他函数，则可以看到代码，甚至可以修改代码。表 2-2 中列出了一些常用常数的值。

表 2-2　常用常数的表示和值

| 常 数 | 值 |
|---|---|
| pi | 3.14159265... |
| i | 虚数单位，$\sqrt{-1}$ |
| j | 与 i 相同 |
| eps | 浮点相对精度，$\varepsilon = 2^{-52}$ |
| realmin | 最小浮点数，$2^{-1022}$ |
| realmax | 最大浮点数，$(2-\varepsilon)2^{1023}$ |
| Inf | 无限值 |
| NaN | 空值 |

下面是一些表达式的例子。

```
rho = (1+sqrt(5))/2
rho =
```

```
    1.6180
a = abs(3+4i)
a =
     5
z = sqrt(besselk(4/3,rho-i))
z =
   0.3730+ 0.3214i
huge = exp(log(realmax))
huge =
  1.7977e+308
toobig = pi*huge
toobig =
   Inf
```

## 2.2 构造数组

在 MATLAB 中构造数组的方法很简单，只需要用空格或逗号间隔数组元素，然后用方括号括起来就行了。例如，

```
x=[0 2 3 6 7 8]
```

就构造了一个有 6 个元素的数组 *x*。

除了直接构造外，还有一些常用的构造方法，下面介绍两种，即增量法和 linspace 函数法。

### 2.2.1 用增量法构造数组

利用 MATLAB 提供的冒号操作符（first:last）可以生成 $1 \times n$ 的矩阵，即数组。数组中的元素按顺序从 first 一直到 last。默认序列是以增量方式生成的，后面的数比它前面第一个数大 1。例如，

```
A = 10：15
A =
    10    11    12    13    14    15
```

数组不必由正整数组成，它也可以包括负值和小数，例如，

```
A = -2.5：2.5
A =
   -2.5000   -1.5000   -0.5000    0.5000    1.5000    2.5000
```

默认时，MATLAB 创建序列时增量总是 1，即使最后的值不是整数，例如，

```
A = 1：6.3
A =
     1     2     3     4     5     6
```

同样，冒号操作符生成的默认序列总是增序排列，而不是减序排列的。下面试图生成减序排列的数值序列时，结果失败。

```
A = 9 : 1
A =
   Empty matrix: 1-by-0
```

实际上，使用冒号操作符时可以指定增量步长值。可以使用（first:step:last）的格式。例如下面创建一个 10 和 50 之间增量为 5 的数值序列。

```
A = 10 : 5 : 50
A =
     10    15    20    25    30    35    40    45    50
```

增量也可以是小数，例如，下面的例子中增量为 0.2。

```
A = 3 : 0.2 : 3.8
A =
     3.0000    3.2000    3.4000    3.6000    3.8000
```

指定增量为负时，创建减序数值序列。例如，

```
A = 9 : -1 : 1
A =
     9    8    7    6    5    4    3    2    1
```

### 2.2.2　用 linspace 函数构造数组

用 linspace 函数构造数组，需要指定首尾值和元素总个数。基本形式是

```
x=linspace(first,last,num)
```

其中，first,last 和 num 分别为 $x$ 数组的首尾元素和元素个数。例如，

```
x=linspace(0,10,5)
x =
          0    2.5000    5.0000    7.5000    10.0000
```

## 2.3　构造矩阵

MATLAB 中，二维数组称为矩阵。图形图像方面要涉及到大量的矩阵运算，比如，一幅数字图像就是一个矩阵，矩阵中的每个元素表示图像上每个像素的信息。那么针对图像所作的任何操作实质上都是针对矩阵进行的。

### 2.3.1　简单的创建方法

MATLAB 中创建矩阵最简单的方法是使用矩阵创建符号[]。在方括号内输入多个元素可以创建矩阵的一个行，并用逗号或空格把每个元素间隔开，例如，

$row = [E_1, E_2, ..., E_m]$　　　　$row = [E_1\ E_2\ ...\ E_m]$

键入下面的代码，创建一个五元素的行。

```
A = [12 62 93 -8 22];
```

如果想开始一个新行，用分号终止当前行。例如，

$A = [row_1; row_2; ...; row_n]$

下面创建一个 3 行 5 列的数值矩阵。注意，所有行必须具有相同的元素个数。

```
A = [12 62 93 -8 22; 16 2 87 43 91; -4 17 -72 95 6]
A =
    12    62    93    -8    22
    16     2    87    43    91
    -4    17   -72    95     6
```

方括号只能创建二维矩阵，包括 0×0，1×1 和 1×$n$ 矩阵。

## 2.3.2 构造特殊矩阵

MATLAB 提供了多个创建不同矩阵的函数，如表 2-3 所示。利用这些函数可以创建各种特殊矩阵。

**表 2-3 特殊矩阵构造函数**

| 函 数 | 功 能 |
|---|---|
| ones | 创建一个所有元素都为 1 的矩阵 |
| zeros | 创建一个所有元素都为 0 的矩阵 |
| eye | 创建对角线元素为 1，其他元素为 0 的矩阵 |
| accumarray | 将输入矩阵的元素分配到输出矩阵中的指定位置 |
| diag | 根据向量创建对角矩阵 |
| magic | 创建一个方形矩阵，其中行、列和对角线上元素的和相等 |
| rand | 创建一个矩阵或数组，其中的元素为服从均匀分布的随机数 |
| randn | 创建一个矩阵或数组，其中的元素为服从正态分布的随机数 |
| randperm | 创建一个向量(1×$n$ 的矩阵) |

表 2-3 中的大部分函数返回 double 型的矩阵。但是，可以用 ones, zeros 和 eye 函数很容易地生成任何数值类型的基本数组。

要做到这一点，需要将 MATLAB 数据类型名称作为最后一个变量，例如，

```
A = zeros(4, 6, 'uint32')
A =
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

又如，下面的代码创建一个 5×5 魔方矩阵。

```
A = magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

注意，每一行、每一列和每个主对角线上的数值加起来都等于 65。

下面的代码创建元素为服从均匀分布的随机数的矩阵或数组，将每个元素乘以 20。

```
A = rand(5) * 20
A =
    3.8686   13.9580    9.9310   13.2046   14.5423
   13.6445    7.5675   17.9954    6.8394    6.1858
    6.0553   17.2002   16.4326    5.7945   16.7699
   10.8335   17.0731   12.8982    6.8239   11.3614
    3.0175   11.8713   16.3595   10.6816    7.4083
```

下面的代码根据向量创建一个对角矩阵。可以将向量元素放在矩阵的主对角线上，或者放在主对角线的上方或下方，如下所示，其中-1 表示将向量元素放在主对角线下方。

```
A = [12 62 93 -8 22];
B = diag(A, -1)
B =
     0     0     0     0     0     0
    12     0     0     0     0     0
     0    62     0     0     0     0
     0     0    93     0     0     0
     0     0     0    -8     0     0
     0     0     0     0    22     0
```

### 2.3.3 聚合矩阵

矩阵聚合是通过连接一个或多个矩阵来形成一个新的矩阵。符号[]不仅是一个矩阵构造符，它还是一个 MATLAB 聚合运算符。表达式 C=[A B]在水平方向上聚合矩阵 ***A*** 和 ***B***，表达式 C=[A;B]在垂向上聚合它们。

下面的例子通过在垂向上聚合矩阵 ***A*** 和 ***B*** 来构造新矩阵 ***C***。

```
A = ones(2, 5) * 6;         % 2×5 的矩阵，元素为 6
B = rand(3, 5);             % 3×5 的矩阵，元素为随机数
C = [A; B]                  % 垂向聚合 A 和 B
C =
    6.0000    6.0000    6.0000    6.0000    6.0000
    6.0000    6.0000    6.0000    6.0000    6.0000
    0.6154    0.7382    0.9355    0.8936    0.8132
    0.7919    0.1763    0.9169    0.0579    0.0099
    0.9218    0.4057    0.4103    0.3529    0.1389
```

可以用聚合的方法创建矩阵甚至多维数组，但不能生成不规则的形状，必须是矩形的。如果是水平生成矩阵，则每个子矩阵必须具有相同的行数；如果是垂向生成矩阵，则每个子矩阵必须具有相同的列数。

图 2-2 显示了两个具有相同高度的矩阵在水平方向上组合成一个新的矩阵。

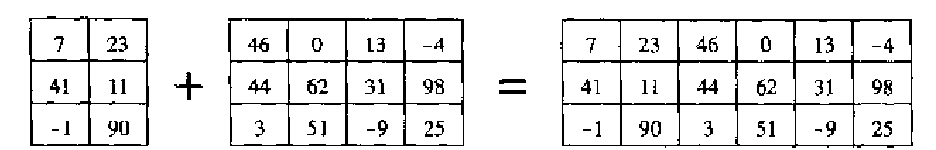


图 2-2　聚合具有相同高度的矩阵

图 2-3 演示了试图水平组合两个不同高度的矩阵的情况。MATLAB 不允许这样做。

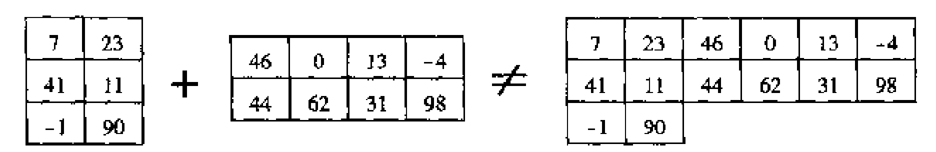


图 2-3　聚合不同高度的矩阵

利用表 2-4 中的函数可以将多个矩阵组合成一个新的矩阵。

表 2-4　矩阵聚合函数

| 函　数 | 描　述 |
|---|---|
| cat | 沿指定的维聚合矩阵 |
| horzcat | 水平聚合矩阵 |
| vertcat | 垂向聚合矩阵 |
| repmat | 通过复制和叠置矩阵来创建新矩阵 |
| blkdiag | 用已有矩阵创建块对角矩阵 |

下面几个例子演示表 2-4 中函数的使用。

使用 cat, horzcat 和 vertcat 函数可以代替[]符号实现矩阵的聚合。下面的两个语句都可以实现命令 C=[A;B]的效果。

```
C = cat(1, A, B);         % 沿第一维聚合
C = vertcat(A, B);        % 垂向聚合
```

用 repmat 函数可以利用已有矩阵的多个拷贝来创建矩阵。键入下面的命令行

```
repmat(M, v, h)
```

MATLAB 将矩阵 ***M*** 在垂向上复制 *v* 次，在水平方向上复制 *h* 次。例如，下面将已有矩阵 ***A*** 复制到新矩阵 ***B*** 中。

```
A = [8 1 6; 3 5 7; 4 9 2]
A =
   8   1   6
   3   5   7
   4   9   2
B = repmat(A, 2, 4)
B =
   8   1   6   8   1   6   8   1   6   8   1   6
   3   5   7   3   5   7   3   5   7   3   5   7
   4   9   2   4   9   2   4   9   2   4   9   2
   8   1   6   8   1   6   8   1   6   8   1   6
```

```
    3   5   7   3   5   7   3   5   7   3   5   7
    4   9   2   4   9   2   4   9   2   4   9   2
```

用 blkdiag 函数创建块对角矩阵，如下所示。

```
A = magic(3);
B = [-5 -6 -9; -4 -4 -2];
C = eye(2) * 8;
D = blkdiag(A, B, C)
D =
    8   1   6   0   0   0   0   0
    3   5   7   0   0   0   0   0
    4   9   2   0   0   0   0   0
    0   0   0  -5  -6   9   0   0
    0   0   0  -4  -4  -2   0   0
    0   0   0   0   0   0   8   0
    0   0   0   0   0   0   0   8
```

### 2.3.4　组合不同类型的数据

构造矩阵时，如果矩阵的数据类型不同，则 MATLAB 会自动对某些元素进行类型转换，然后生成的矩阵具有相同的类型。

用一个高精度的矩阵和一个低精度的矩阵构造新矩阵时，新矩阵是低精度型的。例如，聚合 double 型和 single 型的矩阵时，总是生成 single 型的矩阵。MATLAB 会先将 double 型元素转换为 single 型。

如果用空矩阵参与构造矩阵，则新矩阵中空矩阵被忽略。例如，

```
A = [5.36; 7.01; []; 9.44]
A =
    5.3600
    7.0100
    9.4400
```

下面是几个不同类型矩阵聚合的例子。

（1）single 型和 double 型矩阵聚合。

```
x = [single(4.5) single(-2.8) pi 5.73*10^300]
x =
    4.5000    -2.8000    3.1416       Inf
class(x)                % 显示 x 的数据类型
ans =
   single
```

（2）integer 型和 double 型矩阵聚合。

```
x = [int8(21) int8(-22) int8(23) pi 45/6]
x =
    21    -22    23    3    7
```

```
class(x)
ans =
    int8
```

（3）character 型和 double 型矩阵聚合。

```
x = ['A' 'B' 'C' 68 69 70]
x =
    ABCDEF
class(x)
ans =
    char
```

（4）logical 型和 double 型矩阵聚合。

```
x = [true false false pi sqrt(7)]
x =
    1.0000         0         0    3.1416    2.6458
class(x)
ans =
    double
```

## 2.4 获取矩阵的元素

利用编号和索引，可以获取 MATLAB 矩阵的元素。

### 2.4.1 获取单个元素

要引用矩阵中的特殊元素，用下面的语法指定它的行号和列号，其中 ***A*** 是矩阵变量。按先行后列的顺序指定。

```
A(row, column)
```

例如，对于 4×4 的魔方矩阵 ***A***，

```
A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

下面获取第 4 行第 2 列处的元素。

```
A(4, 2)
ans =
    14
```

### 2.4.2 线性索引

使用 MATLAB，可以用单个编号引用矩阵中的元素。MATLAB 保存矩阵中的数据时不

是按照它们显示在 MATLAB 命令窗口中的形状保存的，而是作为单一元素列保存。这个元素列又是由矩阵中的所有列组成的，后一列元素按先后顺序添加到前一列元素的最后。所以，矩阵 ***A***，

```
A = [2 6 9; 4 2 8; 3 0 1]
A =
      2      6      9
      4      2      8
      3      5      1
```

在内存中是作为下面的序列保存的。

```
2, 4, 3, 6, 2, 5, 9, 8, 1
```

矩阵 ***A*** 第 3 行第 2 列的元素可以看成实际保存序列中的第 6 个元素。要获取这个元素，可以使用标准语法 A(3,2)或使用 A(6)。使用线性索引方式时，如果矩阵大小为[d1 d2]，即 d1 行 d2 列，则位置(i,j)处的元素在保存序列中的位置为

```
(j-1) * d1 + i
```

比如上面矩阵 ***A*** 的大小为[3 3]，则位置(3,2)处的元素在保存序列中的位置为(2-1)*3+3，即 6。

如果知道行-列编号，但是想用线性索引，则可以用 sub2ind 函数进行转换。下面将前面矩阵 ***A*** 的行列索引(3,2)转化为线性索引 6。

```
A = [2 6 9; 4 2 8; 3 0 1];
linearindex = sub2ind(size(A), 3, 2)
linearindex =
     6
```

同样，用 ind2sub 函数根据线性索引值获取行-列索引值。

```
[row col] = ind2sub(size(A), 6)
row =
     3
col =
     2
```

### 2.4.3　获取多个元素

下面 ***A*** 是一个 4×4 的矩阵，利用行-列索引方式可以求得第 4 列元素的和。

```
A = magic(4);
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

可以用冒号操作符和 sum 函数获得上面第二行的结果。

```
sum(A(1:4, 4))
```

下面的语句将矩阵 ***B*** 中的元素每间隔两个元素处的值改为-10。

```
B = A;
B(1:3:16) = -10
B =
   -10      2      3    -10
```

```
    5    11   -10     8
    9   -10     6    12
  -10    14    15   -10
```

利用 end 关键字可以指定矩阵某维的最后一个元素。它适用于不知道矩阵有多少行或多少列的情况。例如，可以用下面的语句进行值的替换。

```
B(1:3:end) = -10
```

利用冒号本身可以引用矩阵某行或列的所有元素。使用下面的语法，可以计算 4×4 的魔方矩阵 **A** 中所有元素的和。

```
sum(A(:, 2))
ans =
    34
```

将冒号用作线性索引，可以引用整个矩阵中的所有元素。本例显示矩阵 **A** 中的所有元素，按列序返回它们。

```
A(:)
ans =
    16
     5
     9
     4
     .
     .
     .
    12
     1
```

## 2.5　获取与矩阵有关的信息

表 2-5 中的函数可以返回关于矩阵形状和大小的信息。

**表 2-5　矩阵信息函数**

| 函　数 | 功　能 |
|---|---|
| length | 返回最长维的长度 |
| ndims | 返回维数 |
| numel | 返回元素个数 |
| size | 返回每一维的长度 |

下面的例子演示表 2-5 中部分函数的使用。

```
A = rand(5) * 10;
A(4:5, :) = []
A =
    9.5013    7.6210    6.1543    4.0571    0.5789
```

```
    2.3114    4.5647    7.9194    9.3547    3.5287
    6.0684    0.1850    9.2181    9.1690    8.1317

% 计算矩阵 A 中所有元素值的均值
sum(A(:))/numel(A)
ans =
    5.8909

% 查找矩阵中大小在 5 和 7 之间的元素
if ndims(A)  ~= 2
   return
end

[rows cols] = size(A);
for m = 1:rows
   for n = 1:cols
      x = A(m, n);
      if x >= 5 && x <= 7
         disp(sprintf('A(%d, %d) = %5.2f', m, n, A(m,n)))
      end
   end
end
```

返回结果

```
A(1, 3) = 6.15
A(3, 1) = 6.07
```

表 2-6 中的函数检查矩阵中的元素是否属于指定的数据类型。

**表 2-6　数据类型检查函数**

| 函　数 | 功　能 |
|---|---|
| isa | 确定输入数据是否属于给定类型 |
| iscell | 确定输入数据是否属于单元数组 |
| iscellstr | 确定输入数据是否属于字符串单元数组 |
| ischar | 确定输入数据是否属于字符数组 |
| isfloat | 确定输入数据是否属于浮点数组 |
| isinteger | 确定输入数据是否属于整型数组 |
| islogical | 确定输入数据是否属于逻辑数组 |
| isnumeric | 确定输入数据是否属于数值数组 |
| isreal | 确定输入数据是否属于实型值数组 |
| isstruct | 确定输入数据是否属于结构数组 |

下面的代码从向量中找出数值型元素。

```
A = [5+7i 8/7 4.23 39j pi 9-2i];
```

```
for m = 1:numel(A)
    if isnumeric(A(m)) && isreal(A(m))
        disp(A(m))
    end
end
```

返回值为

```
1.1429
4.2300
3.1416
```

表 2-7 中的函数检查矩阵中的元素是否为指定的数据结构。

**表 2-7　数据结构检查函数**

| 函　数 | 功　能 |
|---|---|
| isempty | 确定输入数据是否为空 |
| isscalar | 确定输入数据是否为标量 |
| issparse | 确定输入数据是否为稀疏矩阵 |
| isvector | 确定输入数据是否为向量 |

## 2.6　缩放和重塑矩阵

### 2.6.1　放大矩阵

将数据保存在矩阵以外的元素中时，矩阵大小会自动增加，以便容纳下这个新元素。例如，

```
A = magic(4);
A(4, 7) = 17
A =
    16     2     3    13     0     0     0
     5    11    10     8     0     0     0
     9     7     6    12     0     0     0
     4    14    15     1     0     0    17
```

类似地，可以通过指定一系列矩阵元素来扩展矩阵。下面将 4×4 的矩阵放大为 5×7 的矩阵。

```
A(2:5, 5:7) = 5
A =
    16     2     3    13     0     0     0
     5    11    10     8     5     5     5
     9     7     6    12     5     5     5
     4    14    15     1     5     5     5
     0     0     0     0     5     5     5
```

通过将行或列指定为空数组[]，可以从矩阵中删除行和列。

```
A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

然后，用下面的语句删除 **A** 中的第 2 列。

```
A(:, 2) = []
```

**A** 矩阵变为

```
A =
    16     3    13
     5    10     8
     9     6    12
     4    15     1
```

如果从矩阵中删除单个元素，结果将不再是矩阵。例如，

```
A(1 , 2) = []
```

将生成一个错误。但是，可以用线性索引删除单个元素或一系列元素。它将剩下的元素重塑为一个行向量。

```
A(2:2:10) = []
```

生成

```
A =
    16     9     3     6    13    12     1
```

## 2.6.2　重塑矩阵

利用表 2-8 中的函数可以改变矩阵的形状。

**表 2-8　重塑矩阵的函数**

| 函　数 | 功　能 |
|---|---|
| reshape | 重塑矩阵 |
| rot90 | 旋转矩阵 90 度 |
| fliplr | 沿垂轴翻转矩阵 |
| flipud | 沿水平轴翻转矩阵 |
| flipdim | 沿指定方向翻转矩阵 |
| transpose | 沿主对角线翻转矩阵 |
| ctranspose | 转置矩阵 |

下面举几个例子，说明表 2-8 中函数的使用。

（1）重构矩阵　将一个 3×4 的矩阵重构为 2×6 的。

```
A = [1 4 7 10; 2 5 8 11; 3 6 9 12]
A =
```

```
     1     4     7    10
     2     5     8    11
     3     6     9    12

B = reshape(A, 2, 6)
B =
     1     3     5     7     9    11
     2     4     6     8    10    12
```

（2）转置矩阵　可以用转置函数或转置操作符(.')进行矩阵转置。

```
B = A.'
B =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

利用 ctranspose 函数可以进行矩阵的复数共轭转置。该函数的等价操作符是“'”。

```
A = [1+9i 2-8i 3+7i; 4-6i 5+5i 6-4i]
A =
   1.0000 + 9.0000i   2.0000 - 8.0000i   3.0000 + 7.0000i
   4.0000 - 6.0000i   5.0000 + 5.0000i   6.0000 - 4.0000i

B = A'
B =
   1.0000 - 9.0000i   4.0000 + 6.0000i
   2.0000 + 8.0000i   5.0000 - 5.0000i
   3.0000 - 7.0000i   6.0000 + 4.0000i
```

（3）旋转矩阵　将矩阵旋转 90 度。例如，

```
B = rot90(A)
B =
    10    11    12
     7     8     9
     4     5     6
     1     2     3
```

（4）翻转矩阵　下面按照从左向右的方向翻转矩阵 ***A***。

```
B = fliplr(A)
B =
    10     7     4     1
    11     8     5     2
    12     9     6     3
```

## 2.7 导入数据

计算中常常生成大量的数据，将这些数据严格按照 MATLAB 的矩阵格式进行手工构造，工作量非常大。下面介绍几种导入数据的方法。

### 2.7.1 导入文本数据

计算生成的数据有很多是文本格式的，对于 .dat 和 .txt 等类型的文件，可以用 load 命令导入到 MATLAB 工作空间。

假如有图 2-4 所示的文本文件 data.txt，保存在某个目录下。在当前目录窗口中将该目录设置为当前目录以后，在命令窗口中输入下面的命令：

```
load('data.txt','-ascii');
```



图 2-4 文本数据

文件 data.txt 中的数据被导入到工作空间中，并保存在 data 变量中，该变量名与数据的文件名相同。在命令窗口输入

```
data
```

将显示该变量的值，即

```
data =
     1     3     6
     3     5    10
     9     4     8
```

### 2.7.2 导入 MAT 文件数据

MATLAB 中提供了很多 MAT 格式的数据，并利用它们进行功能演示。例如，有一个名为 wind 的 MAT 文件，它代表了某时穿过北美上空的气流数据。利用它，可以绘制向量图形，如流线图、流锥图等。用 load 命令导入 MAT 文件数据。例如，

```
load wind
```

将 wind 数据载入工作空间。现在可以在命令窗口中使用该数据了。

### 2.7.3　使用 Import Wizard 工具

Import Wizard 工具随 MATLAB 一起安装，在 MATLAB 的“Start”菜单中可以找到。利用 Import Wizard 工具，可以导入文本数据、Excel 数据和图像数据等多种类型的数据。该工具的使用比较简单，按照提示一步一步地进行操作就能完成任务。限于篇幅，这里不作详细介绍，该工具的界面如图 2-5 所示。

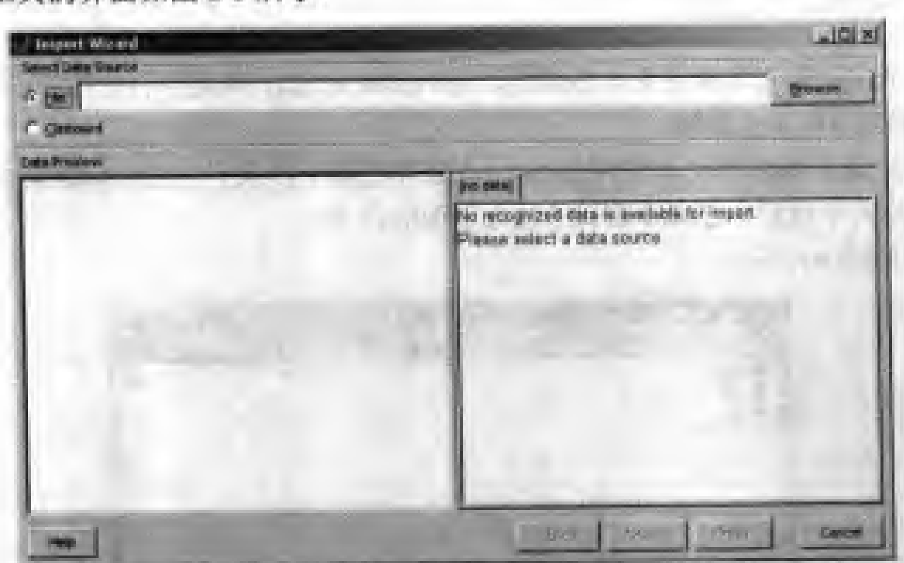



图 2-5　Import Wizard 工具的界面

## 2.8　矩阵的代数运算

在 MATLAB 中，矩阵的基本运算可以像一般数值运算那样进行，也就是说，可以用运算符直接进行计算。

数组和矩阵的加减运算使用加号和减号，即“+”和“-”。例如，

```
x=[3 10 5 6 19];
y=[6 2 9 12 9];
z=x+y
z =
     9    12    14    18    28

x=[5 9;2 7;1 10];
y=[3 11;5 9;5 2];
z=x-y
z =
     2    -2
    -3    -2
    -4     8
```

矩阵相乘使用“*”运算符，例如，

```
x=[1 3;4 6];
y=[6 1;9 3];
z=x*y
z =
     33     10
     78     22
```

如果只是将两个矩阵中相同位置的元素相乘，使用“.*”运算符。例如，将上面的 *x* 和 *y* 进行点乘运算：

```
z=x.*y
```

结果为

```
z =
      6      3
     36     18
```

矩阵除法有左除和右除的区别，分别使用“\”和“/”运算符。右除运算速度要慢一点，而左除运算可以避免奇异矩阵的影响。另外，与“\”和“/”运算符相对应，也有“.\”和“./”运算符，分别用于将两个矩阵中对应元素之间相除。例如，

```
x=[1 3;4 6];
y=[6 1;9 3];
z=x\y
z =
   -1.5000    0.5000
    2.5000    0.1667
z=x.\y
z =
6.0000    0.3333
2.2500    0.5000
```

矩阵与常数的代数运算用得也比较多，可以直接使用上面的各种运算符。例如（对于上面的 *x* 矩阵），

```
z=x+5
z =
      6      8
      9     11
```

## 2.9　矩阵的逻辑运算

合成图像需要用到矩阵的逻辑运算。MATLAB 中的逻辑操作符如表 2-9 所示。

**表 2-9　逻辑操作符**

| 逻辑操作符 | 说　明 | 相对应的函数 |
| --- | --- | --- |
| & | 逻辑与 | and(X,Y) |
| \| | 逻辑或 | or(X,Y) |
| ~ | 逻辑非 | not(X) |
|  | 逻辑异或。*X* 与 *Y* 都是非零或都是零返回 0；有一个非零时返回 1 | xor(X,Y) |

表 2-9 中，函数中的 ***X*** 与 ***Y*** 可以都是矩阵或向量，也可以有一个是标量。要求它们的大小完全相同。例如，

```
x=[9 0 8 2];
y=[8 3 8 0];
z=x & y
z =
     1     0     1     0
x=[8 5 0];
y=2;
z=xor(x,y)
z =
     0     0     1
```

# 第3章 数值计算

MATLAB 具有强大的数值计算功能。本章主要介绍几种常用的计算方法，包括方程求解、多项式计算、插值、基本统计和曲线拟合等。

## 3.1 方程求解

### 3.1.1 求解线性方程组

科学计算中最重要的问题之一就是求解线性方程组。用矩阵表示时，这个问题可以表示为：给定两个矩阵 ***A*** 和 ***B***，总存在一个惟一的矩阵 ***X***，使得 ***AX=B*** 或 ***XA=B***。当矩阵大小为 1×1 时，它就是一个线性方程。

MATLAB 可以用“\”和“/”运算符求解线性方程组，如表 3-1 中所示。

**表 3-1 求解线性方程组**

| | |
|---|---|
| ***X = A\B*** | 求矩阵方程 ***AX = B*** 的解 |
| ***X = B/A*** | 求矩阵方程 ***XA = B*** 的解 |

对于相同的矩阵 ***A*** 和 ***B***，两种运算符之间有下面的关系。

(***B/A***)' = (***A'\B'***)

系数矩阵 ***A*** 的行数和列数不必相等。如果 ***A*** 的大小为 $m×n$，则有 3 种情况，如表 3-2 中所示。

**表 3-2 区分不同系统**

| | |
|---|---|
| $m = n$ | 平方系统，可以求得惟一解 |
| $m > n$ | 超定系统，最少能找到一组解 |
| $m < n$ | 不定系统，解中至多有 $m$ 个非零元素 |

下面根据不同系统的情况介绍线性系统的求解。

**1. 平方系统**

如果矩阵 ***A*** 是非奇异的，则解 ***X=A\B*** 的大小与 ***B*** 的相同，例如，

```
A = pascal(3)
A =
      1     1     1
      1     2     3
      1     3     6
u = [3; 1; 4];
x = A\u
```

```
x =
      10
     -12
       5
```

如果 ***A*** 和 ***B*** 都是方形的，并且大小相等，则解 ***X=A\B*** 的大小与它们相同，例如，

```
B = magic(3)
B =
     8     1     6
     3     5     7
     4     9     2
X = A\B
X =
      19    -3    -1
     -17     4    13
       6     0    -6
```

如果矩阵 ***A*** 的列没有线性相关关系，则它是奇异矩阵。如果 ***A*** 是奇异的，则矩阵方程 ***AX=B*** 的解可能不存在，或者不惟一。使用反斜线运算符“\”时，如果 ***A*** 是近于奇异的，会给出一个警告信息；如果 ***A*** 确实是奇异的，则给出一个错误信息。如果 ***A*** 是奇异的，并且 ***AX=B*** 有解，则通过键入下面的命令行，可以找到多个解。

```
P = pinv(A)*b
```

如果 ***AX=B*** 没有精确解，pinv(A)返回一个最小二乘意义上的解。例如，

```
A = [ 1     3     7
     -1     4     4
      1    10    18 ]
```

是奇异的，可以输入下面的命令行进行确认。

```
det(A)
ans =
     0
```

对于 B=[5;2;12]，方程 ***AX=B*** 有一个精确解，即

```
pinv(A)*b
ans =
     0.3850
    -0.1103
     0.7066
```

输入下面的命令行，可以确认 pinv(A)*B 是一个精确解。

```
A*pinv(A)*b
ans =
     5.0000
     2.0000
    12.0000
```

另一方面，如果 B=[3;6;0]，则 ***AX=B*** 没有精确解。此时，pinv(A)*B 返回一个最小二乘

意义上的解，即

```
A*pinv(A)*b
ans =
   -1.0000
    4.0000
    2.0000
```

2. **超定系统**

由线性方程组组成的超定系统在不同类型的曲线拟合中经常遇到。在下例的数据中，$y$ 是不同时间 $t$ 测得的数据，利用该数据进行曲线拟合，曲线模型假定为

$$y(t) = c_1 + c_2 e^{-t}$$

$t$　0.00 0.30 0.80 1.10 1.60 2.30

$y$　0.82 0.72 0.63 0.60 0.55 0.50

在 MATLAB 命令窗口中用下面的语句输入数据。

```
t = [0 .3 .8 1.1 1.6 2.3]';
y = [.82 .72 .63 .60 .55 .50]';
```

利用最小二乘法拟合，可以计算拟合模型中的待定系数。现在利用给定的数据可以得到 6 个方程，有 2 个未知数，用下面的 6×2 的矩阵表示。

```
E = [ones(size(t)) exp(-t)]
E =
      1.0000    1.0000
      1.0000    0.7408
      1.0000    0.4493
      1.0000    0.3329
      1.0000    0.2019
      1.0000    0.1003
```

使用反斜线运算符获取最小二乘解。例如，

```
c = E\y
c =
      0.4760
      0.3413
```

所以最后得到的曲线模型为

$$y(t) = 0.4760 + 0.3413e^{-t}$$

首先，生成均匀间隔的时间序列 $\boldsymbol{T}$，然后，利用上面得到的模型计算 $\boldsymbol{Y}$ 值，并利用 $\boldsymbol{T}$ 和 $\boldsymbol{Y}$ 绘图。

```
T = (0:0.1:2.5)';
Y = [ones(size(T)) exp(-T)]*c;
plot(T,Y,'-',t,y,'o')
```

通过检验可以发现，$\boldsymbol{E}*\boldsymbol{c}$ 并不精确等于 $\boldsymbol{y}$，但是差距很小，如图 3-1 所示。

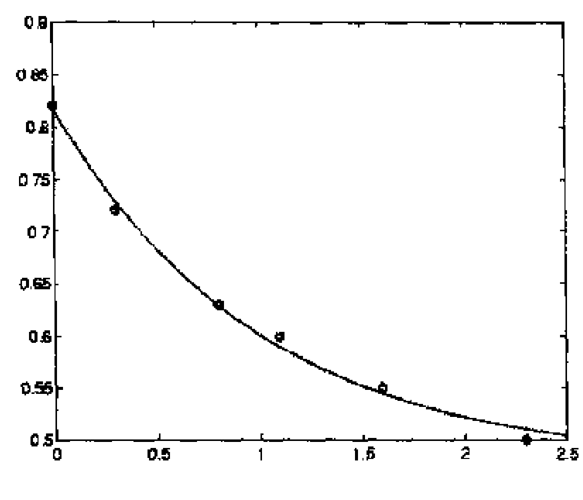



图 3-1　拟合模型的图形

### 3．不定系统

不定系统中，未知数比方程数要多。下面举几个不定系统求解的例子。

```
R = fix(10*rand(2,4))
R =
        6     8     7     3
        3     5     4     1
b = fix(10*rand(2,1))
b =
        1
        2
```

线性系统 ***Rx=b*** 有 2 个方程，4 个未知数。因为系数矩阵包含小整数，可以用 format 命令按有理格式显示解。

```
format rat
p = R\b
p =
       0
     5/7
       0
   -11/7
```

第一个非零元素是 ***p***(2)，因为 ***R***(:,2)是 ***R*** 中范数最大的列。另一个非零元素是 ***p***(4)，因为 ***R***(:,2)被剔除以后，***R***(:,4)最显著。

## 3.1.2　乔累斯基、LU 和 QR 分解

MATLAB 线性方程组的求解是基于 3 种矩阵分解进行的，即乔累斯基分解、LU 分解和 QR 分解，分别用 chol,lu 和 qr 函数实现它们。所有这 3 种分解都使用三角矩阵。三角矩阵中对角线上方或下方的元素都为零。

**1．乔累斯基分解**

乔累斯基分解用三角矩阵和它转置的乘积的形式表示一个对称矩阵，即

$$A=R'R$$

式中，$R$ 是上三角矩阵。

并不是所有的对称矩阵都可以用这种方法进行分解，可以这样分解的矩阵是正定的。这意味着 $A$ 的所有对角线元素都是正的，并且对角线以外的元素都“不太大”。MATLAB 的 Pascal 矩阵提供了一个有趣的示例，后面的例子都使用该矩阵。

```
A = pascal(6)
A =
     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

$A$ 的元素是 binomial 系数，每个元素是北面和西面相邻元素的和。乔累斯基分解为

```
R = chol(A)
R =
     1     1     1     1     1     1
     0     1     2     3     4     5
     0     0     1     3     6    10
     0     0     0     1     4    10
     0     0     0     0     1     5
     0     0     0     0     0     1
```

这些元素又都是 binomial 系数。

乔累斯基分解允许线性系统

$$Ax=b$$

替换为

$$R'Rx=b$$

的形式。因为反斜线运算符能识别三角系统，可以用下面的表达式快速计算。

$$x=R\backslash(R'\backslash b)$$

如果 $A$ 是 $n\times n$ 的，chol(A)的计算复杂度是 $O(n^3)$，但用后面的反斜线求解时复杂度只是 $O(n^2)$。

**2．LU 分解**

LU 分解将任何方形矩阵 $A$ 表示为一个下三角矩阵和一个上三角矩阵的乘积，即

$$A=LU$$

式中，$L$ 是下三角矩阵，对角线元素为 1；$U$ 是上三角矩阵。

矩阵 $A$ 的 LU 分解允许线性系统

$A*x=b$

用下面的表达式快速求解。

$x = U\backslash(L\backslash b)$

下面以 Pasical 矩阵为例，说明 LU 分解的应用。

```
A=pascal(3)
A =
     1     1     1
     1     2     3
     1     3     6
[L,U]=lu(A)
L =
    1.0000         0         0
    1.0000    0.5000    1.0000
    1.0000    1.0000         0
U =
    1.0000    1.0000    1.0000
         0    2.0000    5.0000
         0         0   -0.5000
```

### 3. QR 分解

如果 $Q$ 是正交矩阵，那么它满足

$$Q'Q = 1$$

二维坐标旋转变换矩阵就是一个简单的正交矩阵。

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

正交分解或者说 QR 分解将任意矩形矩阵表示成一个正交矩阵和一个上三角矩阵的乘积，即

$$A = QR$$

下面以 Pascal(3)矩阵 $A$ 为例，用 qr 函数进行 QR 分解。

```
[Q,R]=qr(A)
Q =
   -0.5774    0.7071    0.4082
   -0.5774    0.0000   -0.8165
   -0.5774   -0.7071    0.4082
R =
   -1.7321   -3.4641   -5.7735
         0   -1.4142   -3.5355
         0         0    0.4082
```

## 3.1.3 特征值

假设方形矩阵 $A$ 的特征值和特征向量为标量 $\lambda$ 和非零向量 $v$，则有

$$Av = \lambda v$$

### 1. 特征值分解

对于对角矩阵 $\boldsymbol{\Lambda}$ 对角线上的特征值和组成矩阵 $\boldsymbol{V}$ 的列的对应特征向量，有

$$\boldsymbol{AV}=\boldsymbol{V\Lambda}$$

如果 $\boldsymbol{V}$ 是非奇异的，则有下面的特征值分解形式。

$$\boldsymbol{A}=\boldsymbol{V\Lambda V}^{-1}$$

**例**

```
A =[ 0    -6    -1
     6     2   -16
    -5    20   -10 ];
          lambda = eig(A)
lambda =
        -3.0710
        -2.4645+17.6008i
        -2.4645-17.6008i
```

### 2. Schur 分解

MATLAB 高级矩阵分解不需要特征值分解，而是基于 Schur 分解进行的。Schur 分解可用下式表示。

$$\boldsymbol{A}=\boldsymbol{USU}^{\mathrm{T}}$$

式中，$\boldsymbol{U}$ 是一个正交矩阵，$\boldsymbol{S}$ 是块上三角矩阵，对角线上块的大小为 1×1 和 2×2。

**例**

```
A = [ 6    12    19
     -9   -20   -33
      4     9    15 ]
[U,S] = schur(A)
U =
    -0.4741    0.6648    0.5774
     0.8127    0.0782    0.5774
    -0.3386   -0.7430    0.5774
S =
    -1.0000   20.7846  -44.6948
          0    1.0000   -0.6096
          0         0    1.0000
```

$\boldsymbol{S}$ 矩阵下面的 2×2 块中有两个特征值。

## 3.2 多项式

MATLAB 提供了标准多项式操作的函数，例如多项式求解、评价和差分。另外，还提供了高级应用的函数，例如曲线拟合。多项式函数位于 MATLAB 的 polyfun 目录下。各函数如表 3-3 所示。

表 3-3　多项式函数

| 函　数 | 描　述 |
| --- | --- |
| conv | 多项式相乘 |
| deconv | 多项式相除 |
| poly | 用多项式的根求多项式系数 |
| polyder | 多项式求导 |
| polyfit | 多项式曲线拟合 |
| polyval | 多项式评价 |
| polyvalm | 矩阵多项式评价 |
| residue | 残差运算 |
| roots | 多项式求根，是 poly 函数的逆运算 |

MATLAB 用行向量表示多项式，行向量的元素为多项式的系数，按幂次降序排列。例如，多项式

$$p(x) = x^3 - 2x - 5$$

MATLAB 用下面的向量表示这个多项式。

```
p = [1 0 -2 -5];
```

### 3.2.1　多项式求根

用 roots 函数计算多项式的根，例如，

```
r = roots(p)
r =
      2.0946
     -1.0473 +        1.1359i
     -1.0473 -        1.1359i
```

按照惯例，MATLAB 将根保存在列向量中。函数 poly 返回多项式的系数。

```
p2 = poly(r)
p2 =
1    8.8818e-16    -2    -5
```

poly 和 roots 互为逆函数，可以用它们校验排序、比例化和圆整导致的错误。

用 poly 函数还可以计算矩阵特征多项式的系数，例如，

```
A = [1.2 3 -0.9; 5 1.75 6; 9 0 1];
poly(A)
ans =
     1.0000    -3.9500    -1.8500   -163.2750
```

用 roots 函数算得的这个多项式的根，是矩阵 **A** 的特征根或特征值。可以用 eig 函数直接计算矩阵的特征值。

### 3.2.2　多项式评价

用 polyval 函数计算多项式在指定点上的值。例如，计算 s=5 时 $p$ 的值，使用下面的语句。

```
polyval(p,5)
ans =
   110
```

也可以在矩阵意义上计算多项式的值。此时，上面的多项式成为

$$p(X)=X^3-2X-5I$$

式中，$\boldsymbol{X}$ 是方形矩阵，$\boldsymbol{I}$ 是单位矩阵。例如，下面创建方形矩阵 $\boldsymbol{X}$ 并计算 $\boldsymbol{X}$ 处多项式 $p$ 的值。

```
X = [2 4 5; -1 0 3; 7 1 5];
Y = polyvalm(p,X)
Y =
   377   179   439
   111    81   136
   490   253   639
```

### 3.2.3　卷积和去卷积

多项式的乘和除对应于卷积和去卷积操作。用 conv 和 deconv 函数实现这两个操作。下面以多项式 $a(s)=s^2+2s+3$ 和 $b(s)=4s^2+5s+6$ 为例，计算它们的乘积。

```
a = [1 2 3]; b = [4 5 6];
c = conv(a,b)
c =
     4    13    28    27    18
```

用去卷积的方法，用 $\boldsymbol{c}$ 除以 $\boldsymbol{a}$。

```
[q,r] = deconv(c,a)
q =
     4     5     6
r =
     0     0     0     0     0
```

### 3.2.4　多项式求导

用 polyder 函数计算任何多项式的导数。下面的语句计算多项式 $p$=[1 0 -2 -5]的导数。

```
q = polyder(p)
q =
     3     0    -2
```

polyder 函数还可以计算多项式积和商的导数。例如，对于下面的多项式 $\boldsymbol{a}$ 和 $\boldsymbol{b}$，

```
a = [1 3 5];
b = [2 4 6];
```

用 polyder 函数计算 ***a*b*** 的导数，此时 polyder 函数只有一个输出变量。

```
c = polyder(a,b)
c =
      8      30      56      38
```

使用有两个输出变量的 polyder 函数，可以计算商 ***a/b*** 的导数。

```
[q,d] = polyder(a,b)
q =
     -2      -8      -2
d =
      4      16      40      48      36
```

### 3.2.5　多项式曲线拟合

用 polyfit 函数计算拟合数据集的多项式在最小二乘意义上的系数，调用形式为

```
p = polyfit(x,y,n)
```

***x*** 和 ***y*** 是包含要拟合的 *x* 和 y 数据的向量，***n*** 是多项式的阶次。例如，对于下面的数据

```
x = [1 2 3 4 5]; y = [5.5 43.1 128 290.7 498.4];
```

要求进行三次多项式拟合。

```
p = polyfit(x,y,3)
p =
    -0.1917    31.5821    -60.3262    35.3400
```

下面在一个更好的范围内计算 polyfit 函数的估计值，并通过绘图进行比较。

```
x2 = 1:.1:5;
y2 = polyval(p,x2);
plot(x,y,'o',x2,y2)
grid on
```

生成图 3-2。

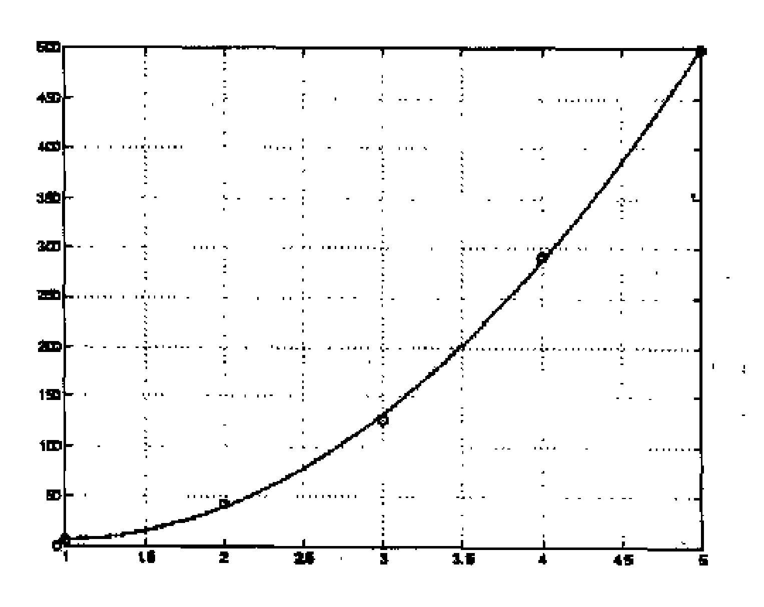


图 3-2　多项式曲线拟合结果

## 3.3 插值

插值方法估计已知数据点之间的值。它在诸如信号和图像处理等方面有着很重要的应用。

MATLAB 提供了很多插值技术，选用时可以在数据拟合的平滑程度、运行速度和内存消耗方面进行平衡。插值函数位于 MATLAB 的 polyfun 目录下。MATLAB 提供的插值函数如表 3-4 所示。

表 3-4 MATLAB 提供的插值函数

| 函 数 | 描 述 |
| --- | --- |
| griddata | 数据网格化和曲面拟合 |
| griddata3 | 三维数据的网格化和超曲面拟合 |
| griddatan | 大于三维数据的网格化和超曲面拟合 |
| interp1 | 一维插值 |
| interp2 | 二维插值 |
| interp3 | 三维插值 |
| interpft | 用快速傅里叶变换(FFT)进行一维插值 |
| interpn | N 维插值 |
| mkpp | 使用分段多项式 |
| pchip | 分段三次 Hermite 插值多项式(PCHIP) |
| ppval | 分段多项式评价 |
| spline | 三次样条数据插值 |
| unmkpp | 分段多项式细节 |

### 3.3.1 一维插值

MATLAB 中有两种一维插值，即多项式插值和基于 FFT 的插值。

**1．多项式插值**

函数 interp1 进行一维插值。一维插值是进行数据分析和曲线拟合的重要手段。interp1 函数使用多项式技术，用多项式函数拟合所提供的数据，并计算目标插值点上的插值函数值，其最常用的语法形式是：

```
yi = interp1(x,y,xi,method)
```

*x* 和 *y* 为给定数据的向量，长度相同。*xi* 为包含要插值的点的向量，method 是一个可选的字符串，指定一种插值方法，包括：

- 最近邻插值(method='nearest') 该方法将插值点的值设置为已知数据点中距离最近的点的值。
- 线性插值(method='linear') 该方法用线性函数拟合每对数据点，并返回 *xi* 处的相关函数值。
- 三次样条插值(method='spline') 该方法用三次样条函数拟合每对数据点，用 spline 函数在插值点处进行三次样条插值。

● 三次插值(method='pchip'或'cubic')　该方法用 pchip 函数对向量 *x* 和 *y* 进行分段三次 Hermite 插值。

这几种方法在速度、内存和平滑性等方面有所差别，在使用时可以根据需要进行选择，包括：

● 最近邻法插值是最快的方法，但是，利用它得到的结果平滑性最差。

● 线性插值比最近邻插值要占用更多的内存，运行时间略长。与最近邻法不同，它生成的结果是连续的，但是在顶点处会有坡度变化。

● 三次样条插值的运行时间相对来说最长，内存消耗比三次插值略少。它生成的结果平滑性最好。但是，如果输入数据不很均匀，可能会得到意想不到的结果。

● 三次插值需要更多内存，而且运行时间比最近邻法和线性插值的长。但是，使用此法时，插值数据及其导数都是连续的。

**2. 基于 FFT 的插值**

函数 interpft 用基于 FFT 的方法进行一维插值。本方法计算包含周期函数值的向量的傅里叶变换。然后，它用更多的点计算逆傅里叶变换。该函数的调用形式为

```
y = interpft(x,n)
```

式中，*x* 是一个包含周期函数值的向量，这些值在等间隔的点上采取。*n* 是样本大小。

### 3.3.2　二维插值

二维插值在图像处理和数据可视化方面有着很重要的应用。MATLAB 用函数 interp2 进行二维插值。该函数的一般形式为

```
ZI = interp2(X,Y,Z,XI,YI,method)
```

其中，**Z** 是一个矩形数组，包含二维函数的值，***X*** 和 ***Y*** 为大小相同的数组，包含相对于 **Z** 的给定值。***XI*** 和 ***YI*** 为包含插值点数据的矩阵，method 表示插值方法，为可选参数。

MATLAB 提供了 3 种不同的插值方法进行二维插值：

● 最近邻插值(method='nearest')　该方法用分段常数曲面拟合数据，插值点的值是最近点的值；

● 双线性插值(method='linear')　该方法用双线性曲面拟合数据点。插值点的值是 4 个最近点的值的组合。本方法是分段双线性的，比双三次插值法快，并且内存消耗更少；

● 双三次插值(method='cubic')　该方法用双三次曲面拟合数据点；插值点的值是 16 个最近点的值的组合，结果的平滑性比前面两种方法都好。

所有这些方法都要求 ***X*** 和 ***Y*** 数据是单调的，即从点到点，要么总是递增的，要么总是递减的。应该用 meshgrid 函数准备这些矩阵。另外，每种方法都会在插值以前自动将输入映射到等间隔的区域。

下面的例子基于一个 7×7 矩阵的数据比较二维插值方法。

（1）生成低分辨率的 peaks 函数图形。

```
[x,y] = meshgrid(-3:1:3);
z = peaks(x,y);
surf(x,y,z)
```

生成图 3-3。

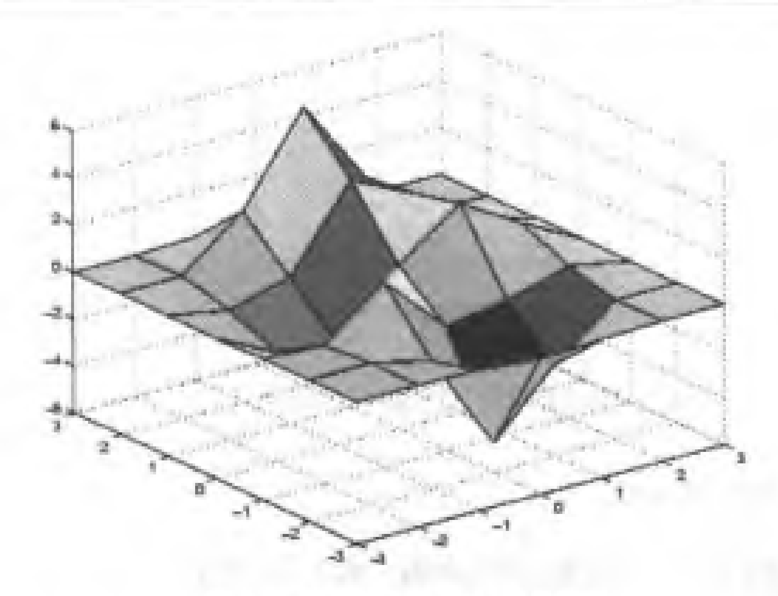


图 3-3 低分辨率的 peaks 函数图形

（2）生成一个精度更高的网格，为插值作准备。

```
[xi,yi] = meshgrid(-3:0.25:3);
```

（3）用最近邻法插值。

```
zi1 = interp2(x,y,z,xi,yi,'nearest');
```

（4）用双线性法插值。

```
zi2 = interp2(x,y,z,xi,yi,'bilinear');
```

（5）用双三次法插值。

```
zi3 = interp2(x,y,z,xi,yi,'bicubic');
```

（6）比较不同插值方法得到的曲面图，如图 3-4 所示。

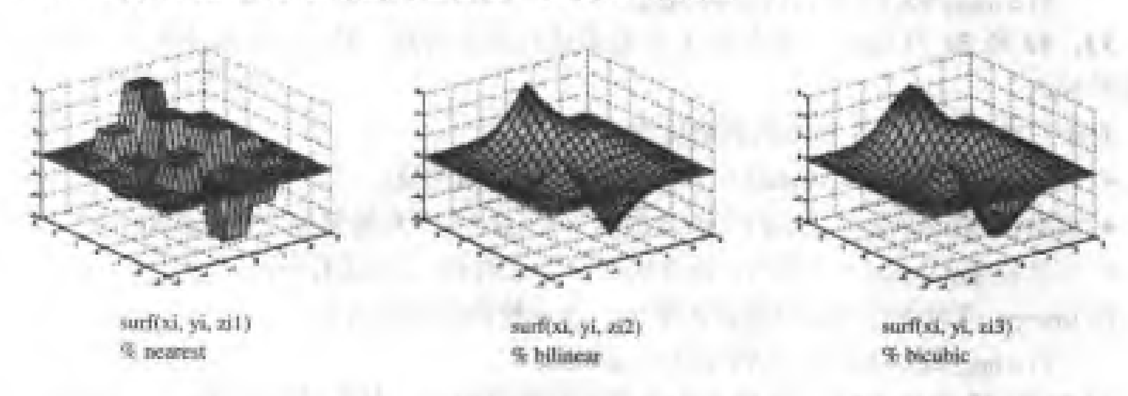
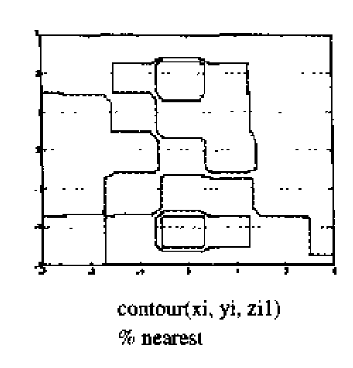
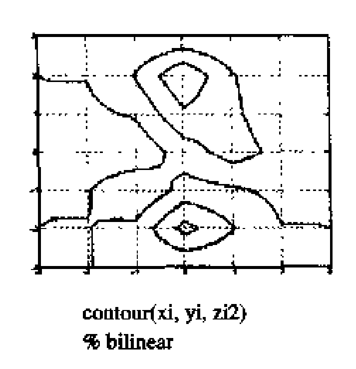



图 3-4 用不同插值方法得到的曲面图

（7）比较不同插值方法得到的等值线图，如图 3-5 所示。


contour(xi, yi, zi1)
% nearest


contour(xi, yi, zi2)
% bilinear

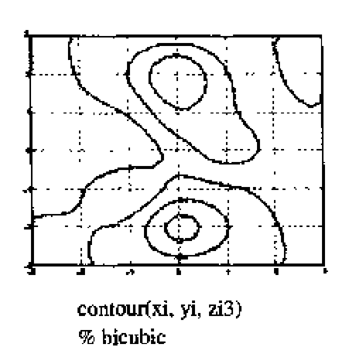

contour(xi, yi, zi3)
% bicubic

图 3-5　用不同方法得到的等值线图

### 3.3.3　插值和多维数组

MATLAB 提供了几个多维数据的插值函数，如表 3-5 所示。

**表 3-5　多维数据的插值函数**

| 函　数 | 描　述 |
|---|---|
| interp3 | 三维数据插值 |
| interpn | 多维数据插值 |
| ndgrid | 多维数据网格化 |

函数 interp3 进行三维插值，计算三维样本集 ***V*** 中数据点之间的值。必须指定一个已知的数据点集：

- ***X***，***Y*** 和 ***Z*** 矩阵指定数据点，它们的值由 ***V*** 给定；
- 矩阵 ***V***，包含与 ***X***，***Y*** 和 ***Z*** 对应的值。

interp3 函数最通用的形式是：

```
VI = interp3(X,Y,Z,V,XI,YI,ZI,method)
```

***XI***，***YI*** 和 ***ZI*** 为 interp3 函数对 ***V*** 中数据进行插值的点。对于超出范围的值，interp3 函数返回 NaN。

对于三维数据，有 3 种不同的插值方法：

- 最近邻法(method = 'nearest')　该方法选择最近点的值；
- 线性插值(method = 'linear')　该方法基于最近的 8 个点进行分段线性插值；
- 三次插值(method = 'cubic')　该方法基于最近的 64 个点进行分段三次插值。

用 interpn 函数进行更高维数据的插值，该函数的常用形式为

```
VI = interpn(X1,X2,X3...,V,Y1,Y2,Y3,...,method)
```

***XI***,***YI*** 和 ***ZI*** 为 interp3 函数对 ***V*** 中数据进行插值的点。对于超出范围的值，interpn 函数返回 NaN。

高维数据插值，同样有最近邻插值、线性插值和三次插值 3 种方法。

用 ndgrid 函数为高维函数评价和插值生成数据数组。该函数将一系列输入向量指定的图域转换为一系列输出数组。第 $i$ 维是输入向量 $x_i$ 的拷贝。

ndgrid 函数的语法格式为

```
[X1,X2,X3,...] = ndgrid(x1,x2,x3,...)
```

例如，假设在给定范围内计算下面三变量函数的值。

$$z = x_2 e^{(-x_1^2 - x_2^2 - x_3^2)}$$

并且有 $-2\pi \le x_1 \le 0.2\pi \le x_2 \le 4\pi$， $0 \le x_2 \le 4\pi$ 。

在命令窗口键入下面的命令行。

```
x1 = -2:0.2:2;
x2 = -2:0.25:2;
x3 = -2:0.16:2;
[X1,X2,X3] = ndgrid(x1,x2,x3);
z = X2.*exp(-X1.^2 -X2.^2 -X3.^2);
slice(X2,X1,X3,z,[-1.2 0.8 2],2,[-2 0.2])
```

生成图 3-6。

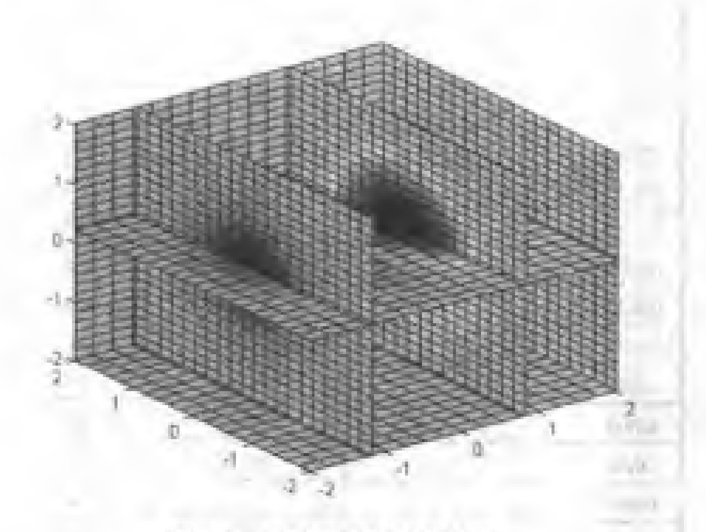



图 3-6 高维数据的插值计算

## 3.4 数据分析和统计

数据分析和统计函数位于 MATLAB 的 datafun 目录下，使用在线帮助可以获得函数的完整列表。

表 3-6 列出的工具箱提供了与数据分析有关的高级功能。

**表 3-6 与数据分析和统计有关的工具箱**

| 工 具 箱 | 数据分析应用 |
|---|---|
| 优化工具箱 | 非线性曲线拟合和回归 |
| 信号处理工具箱 | 信号处理、滤波和频率分析 |
| 样条工具箱 | 曲线拟合和回归 |
| 统计工具箱 | 高级统计分析、非线性曲线拟合和回归 |
| 系统辨识工具箱 | 参数/ARMA 建模 |
| 小波工具箱 | 小波分析 |

### 3.4.1 面向列的数据集合

单变量数据一般保存在单个向量中，向量可以是 $1\times n$ 或 $n\times 1$ 的。对于多变量数据，当然用矩阵表示。但是，用矩阵表示原则上有两个可能的方向，即行方向和列方向。但是，MATLAB 习惯上将不同变量放在列中，观测量放在行中。所以，由 24 个样本组成的 3 变量数据集合用矩阵表示时大小为 24×3。

表 3-7 中是 24 小时内 3 个位置上汽车交通事故的发生次数。

表 3-7　24 小时内 3 个位置上发生的汽车交通事故

| 时　间 | 位置 1 | 位置 2 | 位置 3 |
|---|---|---|---|
| 01h00 | 11 | 11 | 9 |
| 02h00 | 7 | 13 | 11 |
| 03h00 | 14 | 17 | 20 |
| 04h00 | 11 | 13 | 9 |
| 05h00 | 43 | 51 | 69 |
| 06h00 | 38 | 46 | 76 |
| 07h00 | 61 | 132 | 186 |
| 08h00 | 75 | 135 | 180 |
| 09h00 | 38 | 88 | 115 |
| 10h00 | 28 | 36 | 55 |
| 11h00 | 12 | 12 | 14 |
| 12h00 | 18 | 27 | 30 |
| 13h00 | 18 | 19 | 29 |
| 14h00 | 17 | 15 | 18 |
| 15h00 | 19 | 36 | 48 |
| 16h00 | 32 | 47 | 10 |
| 17h00 | 42 | 65 | 92 |
| 18h00 | 57 | 66 | 151 |
| 19h00 | 44 | 55 | 90 |
| 20h00 | 114 | 145 | 257 |
| 21h00 | 35 | 58 | 68 |
| 22h00 | 11 | 12 | 15 |
| 23h00 | 13 | 9 | 15 |
| 24h00 | 10 | 9 | 7 |

首先载入数据。原始数据保存在文件 count.dat 中，格式如下：

```
11    11     9
 7    13    11
14    17    20
11    13     9
43    51    69
38    46    76
61   132   186
```

```
 75   135   180
 38    88   115
 28    36    55
 12    12    14
 18    27    30
 18    19    29
 17    15    18
 19    36    48
 32    47    10
 42    65    92
 57    66   151
 44    55    90
114   145   257
 35    58    68
 11    12    15
 13     9    15
 10     9     7
```

用 load 命令导入数据：

```
load count.dat
```

这将在工作空间中创建一个矩阵。

对于本例，有 3 个变量的 24 个观测量。可以用下面的语句察看。

```
[n,p] = size(count)
n =
    24
p =
     3
```

创建一个时间向量 $t$，元素为从 1 到 $n$ 的整数：

```
t = 1:n;
```

现在根据时间和次数绘图，并标注图形。

```
set(0,'defaultaxeslinestyleorder','-|--|-.')
set(0,'defaultaxescolororder',[0 0 0])
plot(t,count), legend('Location 1','Location 2','Location 3',2)
xlabel('Time'), ylabel('Vehicle Count'), grid on
```

生成的图 3-7 中显示了 24 小时内 3 个位置上发生的交通事故次数。

### 3.4.2 基本数据分析函数

表 3-8 中所示的函数提供了基本的面向列的数据分析能力。这些函数位于 MATLAB 的 datafun 目录下，

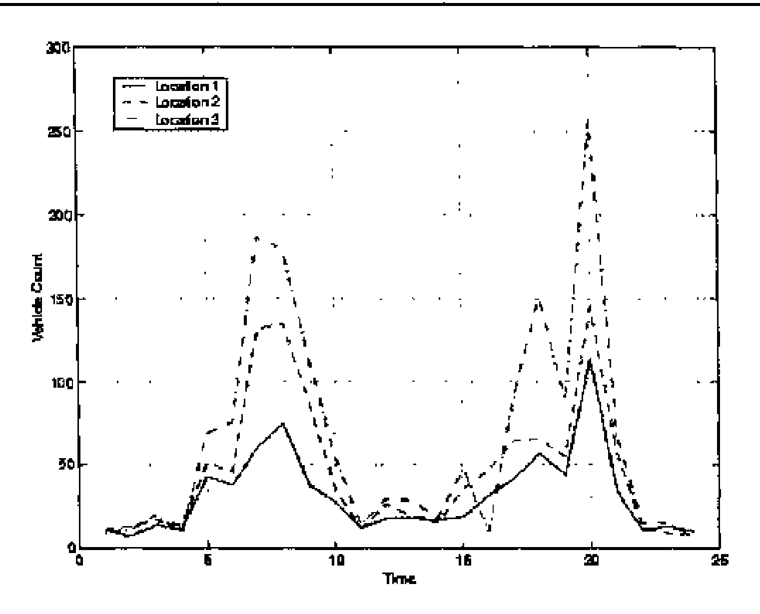



图 3-7　显示数据的线形图

**表 3-8　基本数据分析函数**

| 函　数 | 描　述 |
|---|---|
| cumprod | 元素的累积乘积 |
| cumsum | 元素的累积和 |
| cumtrapz | 累积梯形数值积分 |
| diff | 差分函数和近似求导 |
| max | 最大值 |
| mean | 平均值 |
| median | 中值 |
| min | 最小值 |
| prod | 元素的乘积 |
| sort | 对数组元素按升序或降序排列 |
| sortrows | 将行按升序排列 |
| std | 标准差 |
| sum | 元素的和 |
| trapz | 梯形数值积分 |

使用数据统计工具计算绘图数据的最大值、最小值、均值、中值、极差和标准差，并创建这些统计量的图形。

对于这些函数的向量输入变量，向量方向是行方向还是列方向没有关系。但是，对于数组变量，函数逐列对数组中的数据进行处理。这意味着，例如，将 **max** 函数用于数组，结果是生成一个包含每个列中最大值的行向量。

使用前面汽车交通事故次数示例的数据，下面的语句生成最大值、均值和标准差统计量。

```
mx = max(count)
mu = mean(count)
sigma = std(count)
mx =
```

```
        114         145         257
mu =
      32.0000     46.5417     65.5833
sigma =
      25.3703     41.4057     68.0281
```

要查找这些数据的位置，可以指定第2个输出参数，例如，

```
[mx,indx] = min(count)
mx =
      7     9     7
indx =
      2    23    24
```

说明第1至3个观测点上发生交通事故最少的是02h00,23h00和24h00。

可以从每列数据中减去均值，即

```
[n,p] = size(count)
e = ones(n,1)
x = count - e*mu
```

重置数据可以帮助用户在整个数据集范围内评价向量函数。例如，要找到整个数据集上的最小值，使用下面的语句：

```
min(count(:))
```

生成下面的结果：

```
ans =
      7
```

语法 count(:)将 24×3 的矩阵重置为 72×1 的列向量。

### 3.4.3 方差和相关系数

如表3-9中所示，用cov和corrcoef函数可以计算数据的方差和相关系数。

表3-9 方差和相关系数计算函数

| 函 数 | 描 述 |
|---|---|
| cov | 向量的方差 度量样本数据的范围和离散程度<br>矩阵的方差 度量变量之间线性相关的强度 |
| corrcoef | 相关系数 变量之间线性相关强度的正规化度量 |

**1．方差**

用cov函数返回数据向量的方差。count数据第1列的方差为

```
cov(count(:,1))
ans =
     643.6522
```

对于数据数组，cov函数计算方差矩阵。数组列的方差值沿方差矩阵的对角线放置。剩下的入口反映原始数组各列的方差。对于 $m\times n$ 的矩阵，方差矩阵的大小为 $n\times n$。例如，

$$\begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 \\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_{33}^2 \end{bmatrix}$$

式中，$\sigma_{ij}^2=\sigma_{ji}^2$，$i,j=1,2,3$。

**2. 相关系数**

用 corrcoef 函数生成数据数组的相关系数矩阵，数据数组中行为观测量，列为变量。相关系数是两个变量线性相关关系强度的正规化度量。不相关的数据会导致相关系数为 0，相同数据的相关系数为 1。对于 $m\times n$ 的矩阵，相关系数矩阵的大小为 $n\times n$。相关系数矩阵中元素的放置对应于上面介绍的方差矩阵元素的位置。

对于 count 数据，用下面的语句计算相关系数矩阵。

```
corrcoef(count)
```

生成

```
ans =
    1.0000    0.9331    0.9599
    0.9331    1.0000    0.9553
    0.9599    0.9553    1.0000
```

很清楚，3 个位置上得到的 3 套交通事故发生次数之间有强烈的线性相关关系，因为相关系数接近于 1。

### 3.4.4 有限差分

MATLAB 提供了 3 个函数进行有限差分计算，如表 3-10 所示。

**表 3-10 进行有限差分的相关函数**

| 函　数 | 描　述 |
|---|---|
| diff | 向量连续元素之间的差，向量的数值偏导数 |
| gradient | 矩阵的数值偏导数 |
| del2 | 矩阵的离散拉普拉斯变换 |

diff 函数计算数值向量中连续元素之间的差值。即，diff(X)=[X(2)-X(1)X(3)-X(2)...X(n)-X(n-1)]。所以，对于下面的矩阵 ***A***，有

```
A = [9 -2 3 0 1 5 4];
diff(A)
ans =
    -11     5    -3     1     4    -1
```

除了计算第一个差值以外，diff 函数还可以帮助确定某些向量特性。例如，可以用 diff 函数确定向量是否是单调的（元素总是按升序或降序排列），或者向量中的元素是否是等间隔排列的。表 3-11 介绍了将向量 ***x*** 用于 diff 函数的不同方法。

表 3-11 将向量 x 用于 diff 函数的几种方法

| 使用 diff 函数的不同方法 | 描 述 |
| --- | --- |
| diff(x)==0 | 测试是否有重复元素 |
| all(diff(x)>0) | 测试 x 的单调性 |
| all(diff(diff(x))==0) | 测试向量元素是否等间隔排列 |

### 3.4.5 数据预处理

**1．缺失值**

特殊值 NaN 在 MATLAB 中表示它不是一个值。IEEE 浮点规范将 NaN 指定为未定义的表达式如 0/0 的结果。正确掌握缺失数据的使用是一个困难的问题，并且经常根据不同情况有所变化。如果是为了进行数据分析，通常用 NaN 表示缺失数据或者无法获得的数据。

下面的例子中，考虑一个魔方矩阵，其中心元素设置为 NaN。

```
a = magic(3); a(2,2) = NaN
a =
     8      1      6
     3    NaN      7
     4      9      2
```

为矩阵的每一列数据求和：

```
sum(a)
ans =
    15    NaN     15
```

任何与 NaN 有关的算术运算得到的结果都是 NaN。

进行统计计算以前，应该将 NaN 从数据中剔除。表 3-12 中列出了用 isnan 函数剔除 NaN 的几种方法。

表 3-12 用 isnan 函数剔除 NaN 的几种方法

| 代 码 | 描 述 |
| --- | --- |
| i = find(~isnan(x));<br>x = x(i) | 查找向量中非 NaN 的元素的编号，然后只保留非 NaN 元素 |
| x = x(find(~isnan(x))) | 从向量中剔除 NaN |
| x = x(~isnan(x)); | 从向量中快速剔除 NaN |
| x(isnan(x)) =[]; | 从向量中剔除 NaN |
| X(any(isnan(X)'),:) =[]; | 从矩阵 X 中剔除任何包含 NaN 的行 |

如果经常需要删除 NaN，可以编写一个比较短的 M 文件函数：

```
function X = excise(X)
X(any(isnan(X)'),:) = [];
```

现在键入

```
X = excise(X);
```

可以完成相同的任务。

### 2. 剔除异常值

可以用与处理 NaN 大致相同的方式从数据集中剔除异常值或不匹配数据点。对于前面用到的 count 数据，每一列的均值和标准差为

```
mu = mean(count)
sigma = std(count)
mu =
32.0000    46.5417    65.5833
sigma =
25.3703    41.4057    68.0281
```

用下面的语句获取含有大于三倍标准差的异常值的行数。

```
[n,p] = size(count)
outliers = abs(count - mu(ones(n, 1),:)) > 3*sigma(ones(n, 1),:);
nout = sum(outliers)
nout =
        1    0    0
```

在第一列中有一个异常值。将它从整个观测数据中剔除：

```
count(any(outliers'),:) = [];
```

## 3.4.6 回归分析

在处理科学数据的时候，常常需要描述某些变量之间的关系。如果是线性关系，可以用 MATLAB 的反斜线运算符求回归方程或曲线方程的系数。

假设有下面的对应数据 $t$ 和 $y$，首先绘数据图。

```
t = [0 .3 .8 1.1 1.6 2.3]';
y = [0.5 0.82 1.14 1.25 1.35 1.40]';
plot(t,y,'o'), grid on
```

生成图 3-8。

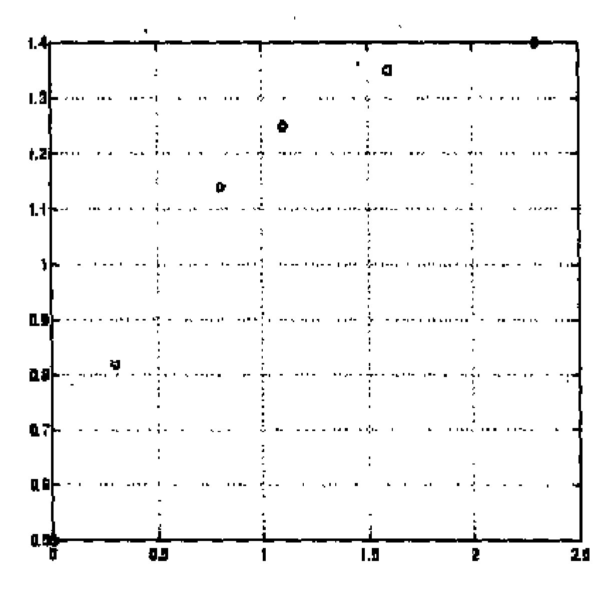


图 3-8　数据散点图

根据图 3-8 中的散点图，大致确定该数据可以用下面的二次多项式拟合。

$$y = a_0 + a_1 t + a_2 t^2$$

未知系数 $a_0$, $a_1$ 和 $a_2$ 可以用最小二乘法进行拟合。该方法使数据与模型之间的均方差最小。这个系统有 6 个方程，3 个未知数，如下所示。

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \\ 1 & t_5 & t_5^2 \\ 1 & t_6 & t_6^2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

可以用一个 6×3 的矩阵表示。

```
X = [ones(size(t))   t   t.^2]
X =
     1.0000         0         0
     1.0000    0.3000    0.0900
     1.0000    0.8000    0.6400
     1.0000    1.1000    1.2100
     1.0000    1.6000    2.5600
     1.0000    2.3000    5.2900
```

用反斜线运算符进行求解。

```
a = X\y
a =
     0.5318
     0.9191
    - 0.2387
```

所以，最后求得的多项式模型为

$$y = 0.5318 + 0.9191t - 0.2387t^2$$

现在计算等间隔点上的模型值，并在图上叠加原始数据。

```
T = (0:0.1:2.5)';
Y = [ones(size(T))   T   T.^2]*a;
plot(T,Y,'-',t,y,'o'), grid on
```

生成结果如图 3-9 所示。

很明显，这个拟合结果还不够理想。可以增加多项式拟合的阶次，或者试着用其他形式的函数来获取更好的近似。

可以试着用线性参数函数来取代多项式函数。本例中，考虑下面的指数函数。

$$y = a_0 + a_1 e^{-t} + a_2 t e^{-t}$$

未知系数 $a_0$, $a_1$,和 $a_2$ 可以用最小二乘法进行拟合。

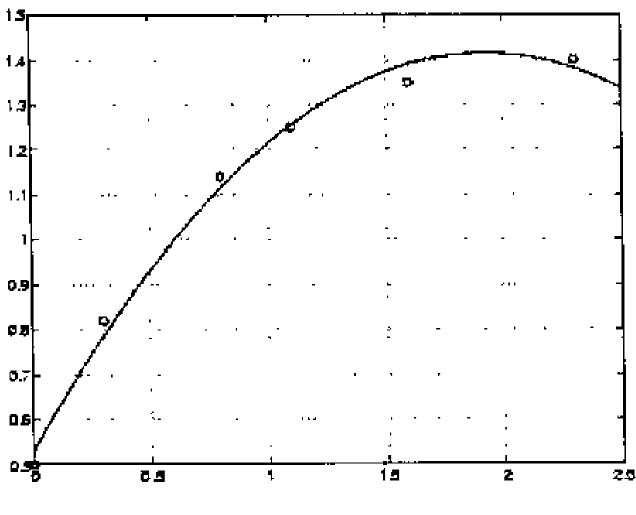



图 3-9　多项式拟合曲线

对于上一小节中的例题数据，首先组成回归矩阵 $\boldsymbol{X}$，然后用反斜线运算符求模型系数。

```
X = [ones(size(t))   exp(-t)   t.*exp(-t)];
a = X\y
a =
      1.3974
     -0.8988
      0.4097
```

所以，最后得到的拟合模型为

$$y = 1.3974 - 0.8988\mathrm{e}^{-t} + 0.4097t\mathrm{e}^{-t}$$

现在计算等间隔点上的模型值，并在图上叠加原始数据点。

```
T = (0:0.1:2.5)';
Y = [ones(size(T))   exp(-T)   T.*exp(-T)]*a;
plot(T,Y,'-',t,y,'o'), grid on
```

生成图 3-10。

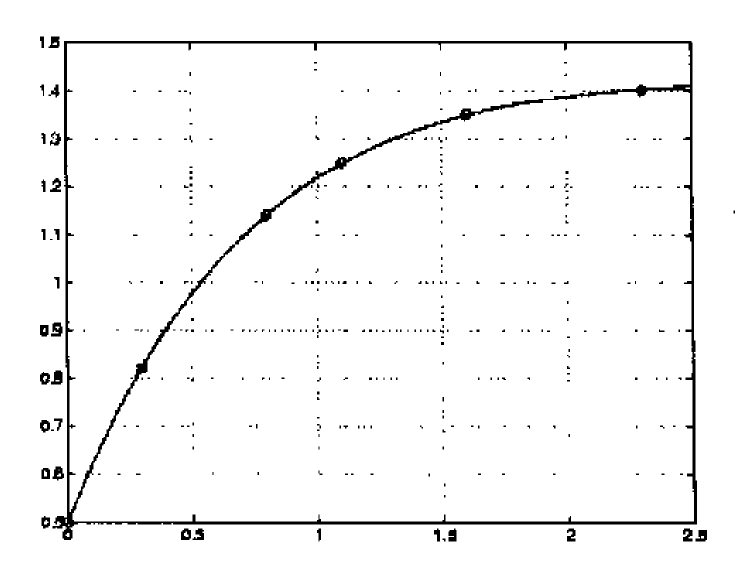



图 3-10　线性参数函数的拟合结果

显然，现在的拟合结果要比二次多项式函数的好得多。

如果 y 是多个独立变量的函数，可以用多元回归的方法建立 y 和各独立变量之间的关系模型。

假设对于参数 x1 和 x2 以及对应的 y，有下面的测量值。

```
x1 = [.2 .5 .6 .8 1.0 1.1]';
x2 = [.1 .3 .4 .9 1.1 1.4]';
y = [.17 .26 .28 .23 .27 .24]';
```

假设数据模型为

$$y = a_0 + a_1x_1 + a_2x_2$$

采用最小二乘拟合，可以求取模型中的待定系数 $a_0$, $a_1$ 和 $a_2$。先构造方程组的系数矩阵 $\boldsymbol{X}$，然后用反斜线运算符求解系数。

```
X = [ones(size(x1))   x1   x2];
a = X\y
a =
     0.1018
     0.4844
    -0.2847
```

所以，数据模型的最小二乘拟合模型为

$$y = 0.1018 + 0.4844x_1 - 0.2847x_2$$

为了知道模型的精度，下面计算数据与模型计算值之间的最大绝对差。

```
Y = X*a;
MaxErr = max(abs(Y - y))
MaxErr =
    0.0038
```

最大绝对差足够小，所以认为模型是合理的。

### 3.4.7　曲线拟合

本小节以个案学习的形式介绍几种 MATLAB 基本数据分析功能，包括多项式拟合、残差分析、指数拟合和误差边界等。

文件 census.mat 中包含了美国 1790 年到 1990 年各年的人口数据，在 MATLAB 命令窗口载入它：

```
load census
```

现在工作空间包括两个新变量，即 cdate 和 pop。cdate 变量是一个列向量，按增量 10 包含了 1790 年到 1990 年的年份；pop 变量也是一个列向量，包含与 cdate 年份对应的美国人口数据。

**1．多项式拟合**

首先用 polyfit 和 polyval 函数（用简单的多项式）拟合 census 数据。MATLAB 的 polyfit 函数生成给定数据集最小二乘意义上指定阶次的最佳拟合多项式。下面进行 4 阶多项式拟合

```
p = polyfit(cdate,pop,4)
```

```
Warning: Polynomial is badly conditioned. Remove repeated data
points or try centering and scaling as described in HELP POLYFIT.
p =
   1.0e+05 *
    0.0000    -0.0000    0.0000    -0.0126 6.0020
```

产生了警告信息是因为矩阵中数据的大小相差太大。处理方法之一是对数据进行正规化或标准化。

首先对数据进行预处理，正规化数据。正规化是一种比例化处理方法，可以改善后面数值计算的精度。正规化数据的一种方法是使数据均值为 0，大小在一个标准差范围内，即

```
sdate = (cdate - mean(cdate))./std(cdate)
```

现在用正规化后的数据拟合 4 阶多项式模型。

```
p = polyfit(sdate,pop,4)
p =
    0.7047    0.9210    23.4706    73.8598    62.2285
```

在正规化后的年份值上计算拟合后的多项式值，并绘制拟合数据和测量数据的图形。

```
pop4 = polyval(p,sdate);
plot(cdate,pop4,'-',cdate,pop,'+'), grid on
```

生成图 3-11。

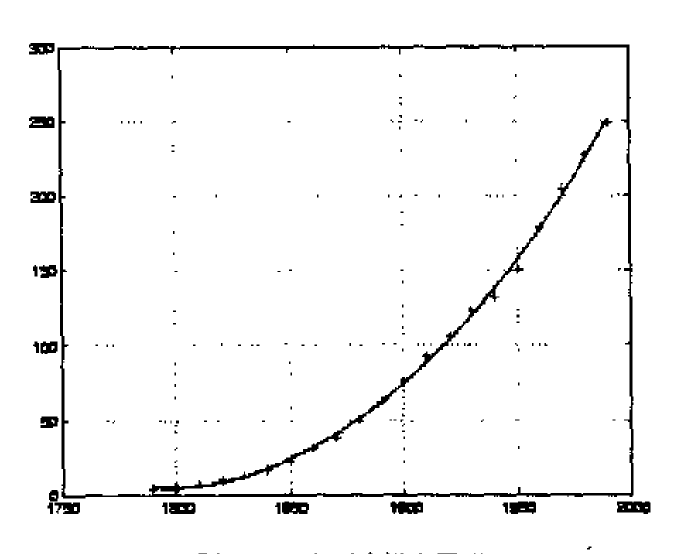


图 3-11　多项式拟合图形

**2．残差分析**

可以用残差度量拟合效果的好坏，残差是测量值与预测值之间的差值。下面用正规化后的 cdate 值比较不同拟合方法得到的残差，分别用线性模型、2 次模型、4 阶模型和指数模型等进行拟合。

（1）用线性模型进行拟合

在命令窗口接着键入下面的命令行。

```
p1 = polyfit(sdate,pop,1);
pop1 = polyval(p1,sdate);
plot(cdate,pop1,'b-',cdate,pop,'g+')
```

生成图 3-12。

键入下面的代码，计算残差并绘残差图，如图 3-13 所示。

```
res1 = pop - pop1;
figure, plot(cdate,res1,'g+')
```

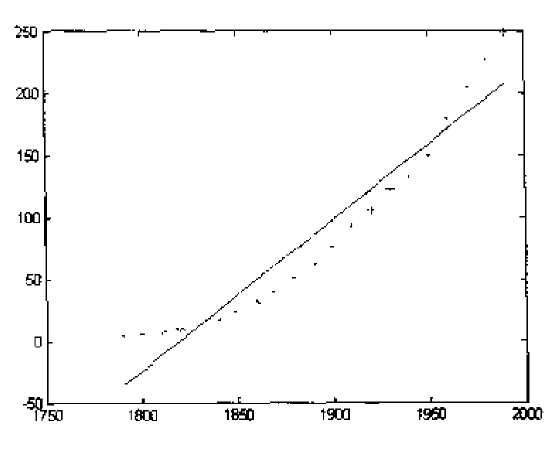


图 3-12　用线性模型拟合

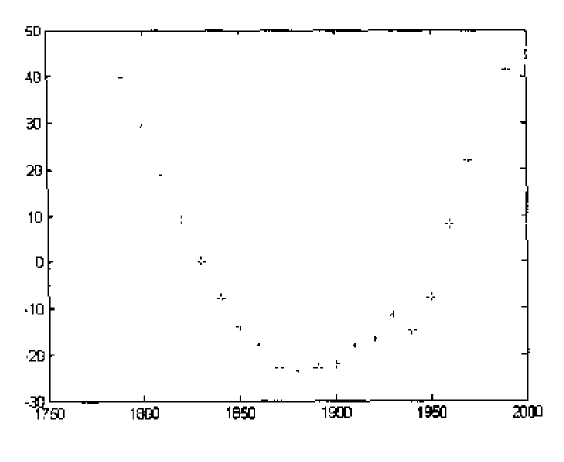


图 3-13　残差图

从图 3-12 中可以看出，线性模型的拟合效果不很理想，而且残差图中的散点呈现明显的规律性（散点呈均匀分布比较好）。所以进一步使用其他模型。

（2）用 2 次模型进行拟合

在命令窗口键入下面的命令行，用 2 次模型拟合数据。

```
p = polyfit(sdate,pop,2);
pop2 = polyval(p,sdate);
plot(cdate,pop2,'b-',cdate,pop,'g+')
```

生成图 3-14。

然后计算原数据和预测数据的残差。

```
res2 = pop - pop2;
figure, plot(cdate,res2,'g+')
```

生成残差图如图 3-15 中所示。

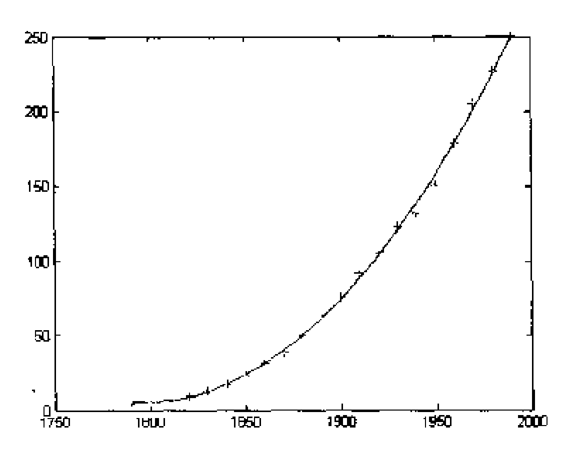


图 3-14　用 2 次模型拟合

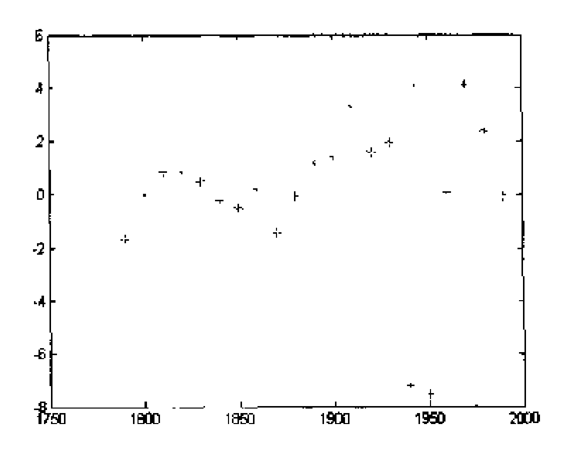


图 3-15　2 次模型拟合的残差图

从图 3-14 和图 3-15 中可以看出，2 次模型的拟合效果要比线性模型的好得多。但残差图仍然呈现强烈的规律性。

（3）用 4 阶模型进行拟合

进一步用 4 阶模型进行拟合，在命令窗口键入下面的命令行。

```
p = polyfit(sdate,pop,4);
pop4 = polyval(p,sdate);
plot(cdate,pop4,'b-',cdate,pop,'g+')
```

生成图 3-16。

计算残差，绘残差图。

```
res4 = pop - pop4;
figure, plot(cdate,res4,'g+')
```

如图 3-17 所示。

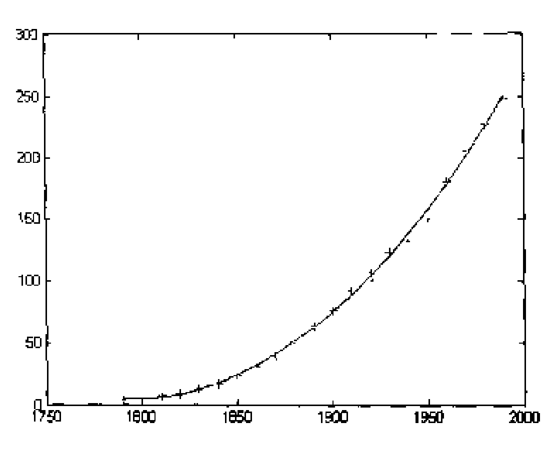





图 3-16　4 次模型的拟合结果

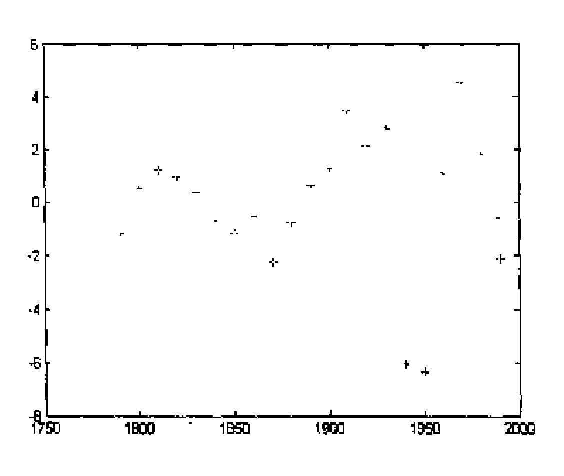



图 3-17　4 次模型的残差图

可见，4 阶模型的拟合效果略有改善，但残差图仍然呈现明显的规律性。

（4）用指数模型进行拟合

从数据图中可以看出，数据曲线呈现一定的指数特征。利用这一点，下面用人口数据的对数值与正规化后的年份值进行拟合。

```
logp1 = polyfit(sdate,log10(pop),1);
logpred1 = 10.^polyval(logp1,sdate);
semilogy(cdate,logpred1,'-',cdate,pop,'+');
grid on
```

生成图 3-18。

下面用 2 次模型进行拟合。

```
logp2 = polyfit(sdate,log10(pop),2);
logpred2 = 10.^polyval(logp2,sdate);
semilogy(cdate,logpred2,'-',cdate,pop,'+'); grid on
```

生成图 3-19。

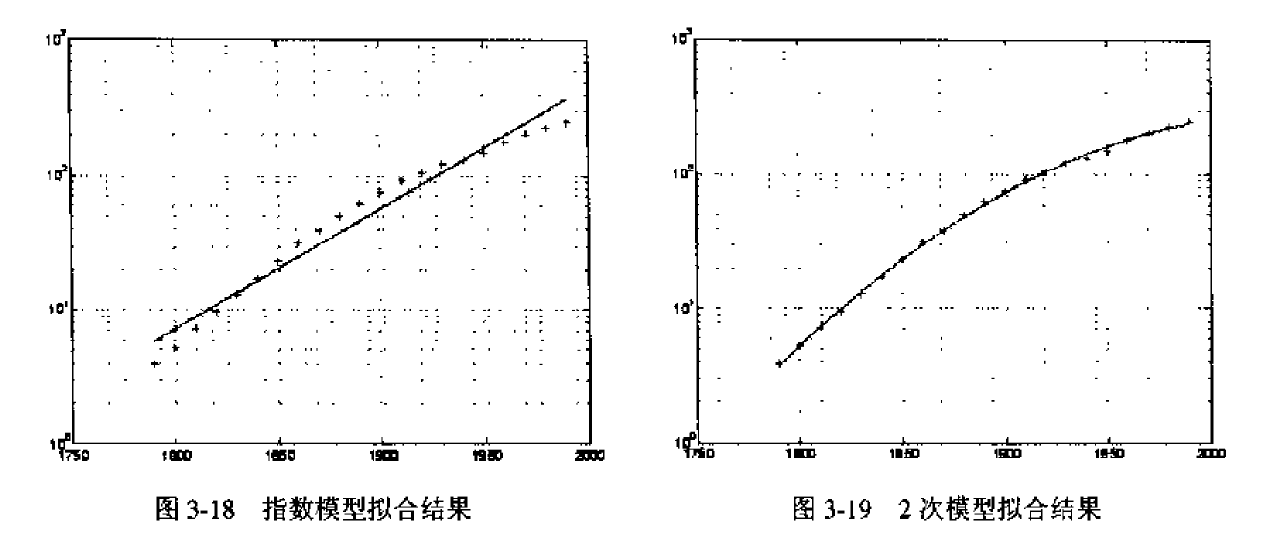


图 3-18 指数模型拟合结果　　图 3-19 2 次模型拟合结果

显然，这是一个更精确的模型。现在比较一下取对数前后分析得到的残差。

在命令窗口键入下面的命令行，计算取对数前的残差，并绘残差散点图。

```
r = pop - 10.^(polyval(logp2,sdate));
plot(cdate,r,'+')
```

生成图 3-20。

键入下面的命令行，计算取对数后的残差，并绘残差图。

```
logres2 = log10(pop)- polyval(logp2,sdate);
plot(cdate,logres2,'+')
```

生成图 3-21。

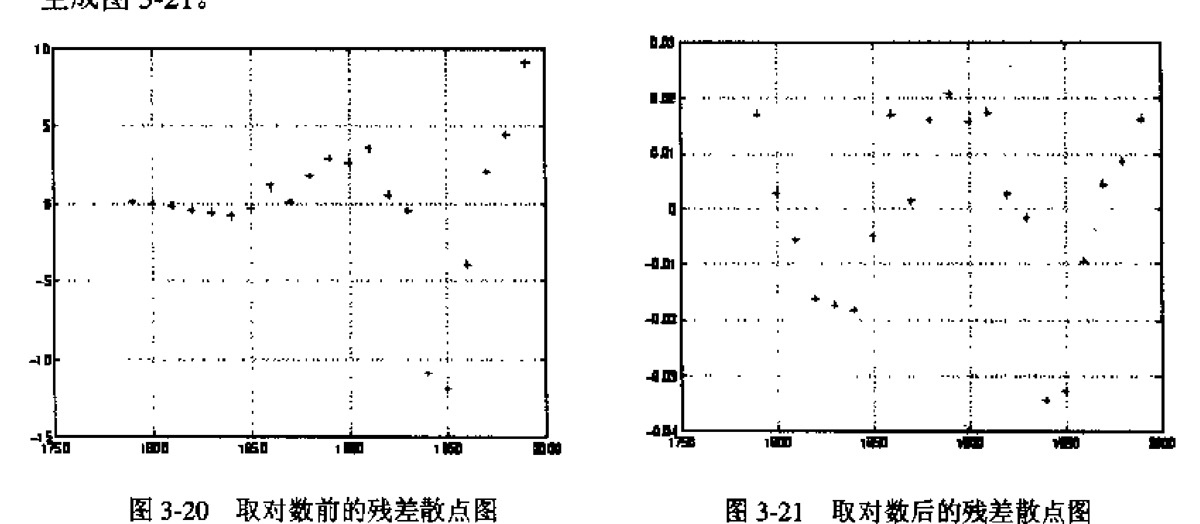


图 3-20 取对数前的残差散点图　　图 3-21 取对数后的残差散点图

显然，取对数以后残差点的分布更具随机性。

### 3. 误差边界

确定数据拟合结果是否合理时可以使用误差边界。将 polyfit 函数的第二个输出参数作为 polyval 的输入参数，可以获取误差边界。

下面的例子继续使用 census 数据并对它进行正规化。然后用 polyfit 和 polyval 函数生成 2 次多项式模型的误差边界。年份值经过了正规化。下面的代码使落在误差边界内的点占总点数的 95%。

```
load census
sdate = (cdate - mean(cdate))./std(cdate)

[p2,S2] = polyfit(sdate,pop,2);
[pop2,del2] = polyval(p2,sdate,S2);
plot(cdate,pop,'+',cdate,pop2,'g-',cdate,pop2+2*del2,'r:',...
    cdate,pop2-2*del2,'r:'), grid on
```

生成图 3-22。

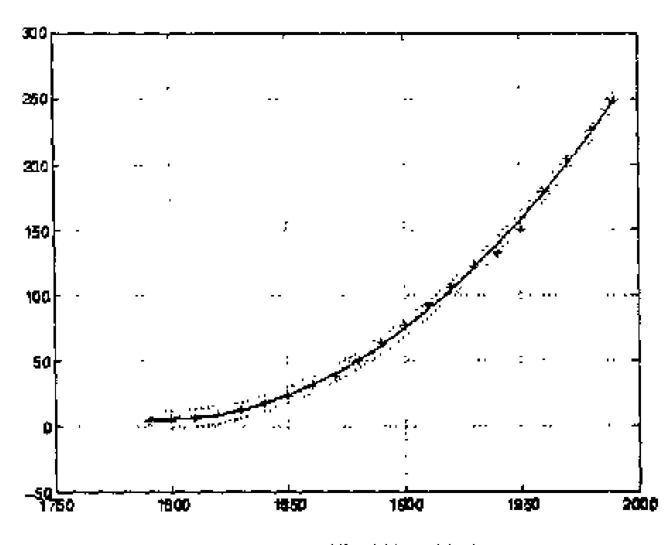


图 3-22　2 次模型的误差边界

**4. 基本拟合界面**

MATLAB 支持用基本拟合界面进行曲线拟合。它具有拟合快速，操作简便的优势。该界面具有如下功能：

- 数据的拟合可以通过插值法、分段 3 次艾尔米特插值（PCHIP）或者是 1 到 10 阶的多项式实现；
- 利用一组数据可以同时作多条拟合曲线；
- 可以绘制残差图；
- 可以检查拟合结果；
- 可以进行拟合曲线的估计；
- 用拟合结果和标准残差来注释；
- 可以保存拟合曲线和计算结果。

用户可以把该界面与命令函数结合起来应用。

按照下面的步骤打开曲线拟合界面。

（1）导入数据；

（2）在“Tools”菜单中单击“Basic Fitting”菜单选项；

（3）双击按钮。

打开的曲线拟合界面“Basic Fitting”对话框，如图 3-23 所示。

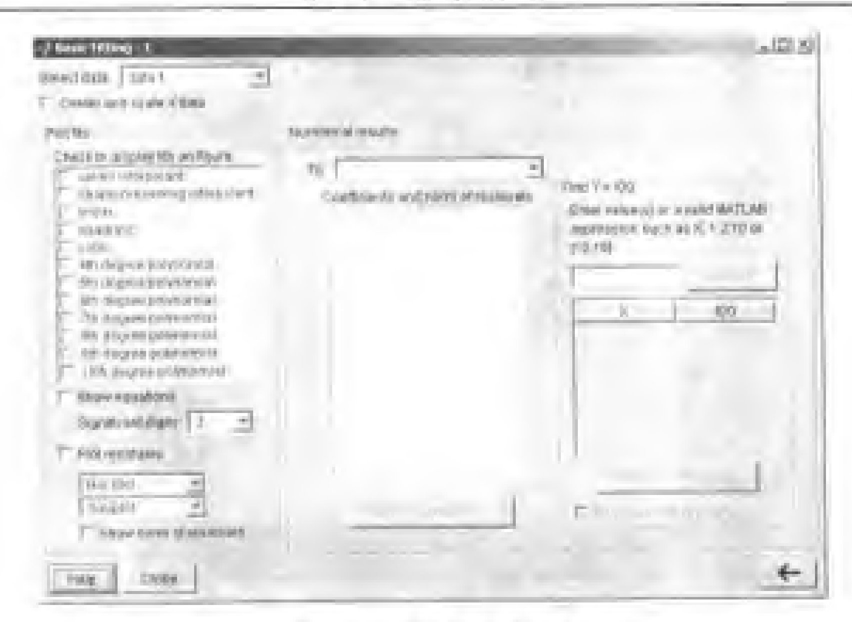


图3-23 基本的拟合界面

基本拟合界面中各选项的功能包括：

- Select data 下拉式列表框 选择所要拟合的数据。一次只能选择一组数据，但在一组数据里可以同时拟合多条曲线。可以用“Plot Editor”改变数据的名称。
- Center and scale x data 单选钮 对数据进行中心化和离散化，数据经过处理后可以提高计算的准确性。
- Plot fit 方框 拟合图形。对于当前数据，这个子菜单提供了多种拟合方法。
- Check to display fits on figure 提供了多种曲线拟合的方法，用户可以选择一个或多个。计算以后，选中的各个方法所对应的拟合曲线显示在图形界面中，可以根据标准残差的大小来选择理想的拟合曲线。一般，标准残差越小，曲线的拟合效果越好。
- Show equation 单选钮 选此项，在拟合图形上显示方程。
- Plot residuals 单选钮 选此项，显示拟合曲线的残差，可用条形图、散点图或线形图显示。
- Show norm of residuals 单选钮 选此项，显示标准残差。
- Numerical results 方框 计算当前拟合图形的参数。
- Fit 选择当前数据集拟合的方程。
- Coefficients and norm of residuals 显示拟合数据集的计算结果。
- Save to workspace 保存计算结果。
- Find Y = f(X) 用内推法或外推法求当前拟合曲线的值。

用基本拟合界面拟合软件自带数据 census.mat，按照下面的步骤进行：

（1）首先建立一个M文件：

```
load census
plot(cdate,pop,'ro')
```

（2）从“Tools”菜单中选择“Basic Fitting”选项，如图3-24中所示。

（3）在基本拟合界面中作以下设置：

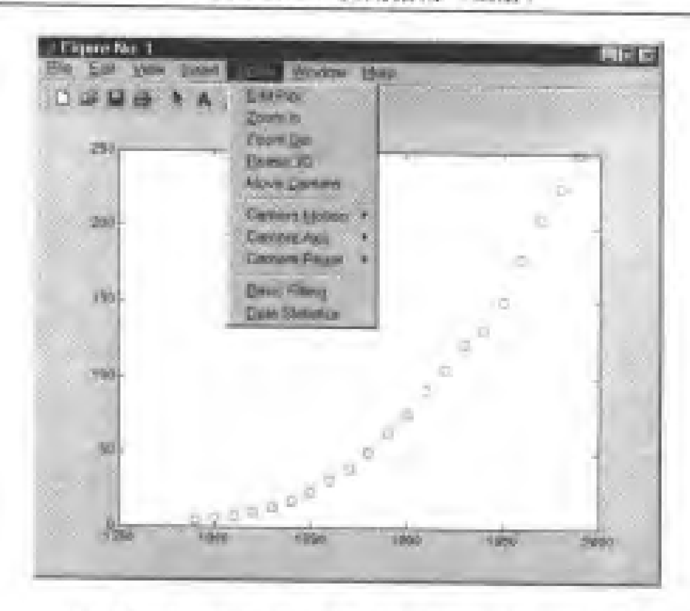


图 3-24　从“Tools”菜单中选择“Basic Fitting”选项

- 用 3 次多项式拟合数据；
- 在拟合图上显示方程；
- 将拟合残差作为条形图显示，并将残差的图形作为子图显示；
- 显示残差的范数。

结果如图 3-25 中所示。




图 3-25　进行选项设置

利用“Plot fits”面板可以可视地探察当前数据集的多个拟合图形。为了进行比较，通过选择合适的核选框来拟合 census 数据的其他方程。如果某个方程生成的结果在数值上不精确，MATLAB 会显示一个警告信息。此时，应该选择“Center and scale X data”核选框来改进数值精度。

最后将拟合结果和残差显示在图 3-26 中。

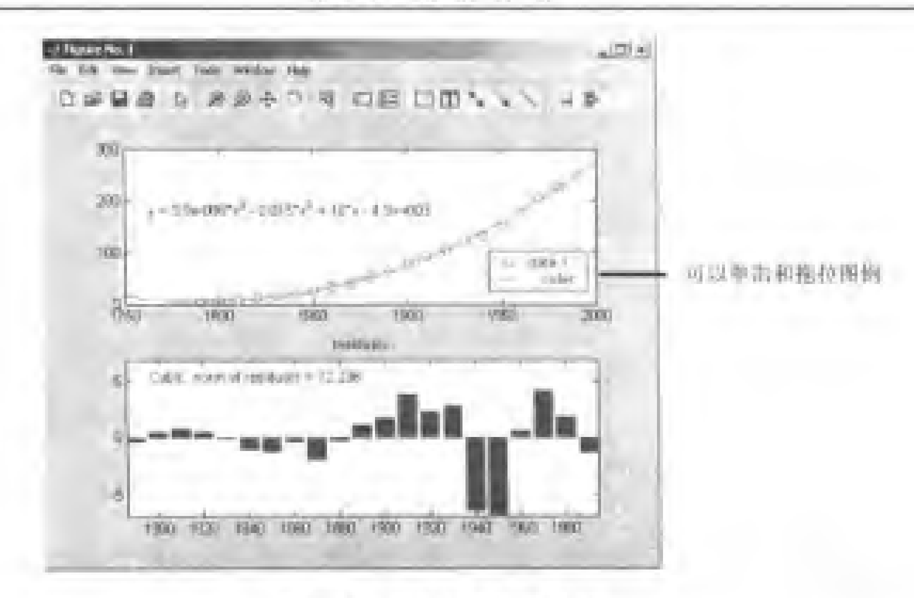



图 3-26 拟合结果和残差图

图例显示了数据集和方程的名称。如果图例覆盖了图形的一部分，可以通过单击和拖拉操作将它移到其他地方。添加和删除数据集或拟合线时图例会自动更新，拟合线用默认的线型和颜色显示。可以用绘图编辑器改变任何默认的图形设置。但是，如果紧接着进行另一个拟合，会取消所做的任何改变。要使改变的设置生效，必须等到数据拟合过程结束。

通过选择➡按钮，可以检查拟合系数和残差范数。

利用"Fit"菜单，可以在没有绘拟合图的情况下探察当前数据集的数值拟合结果。为了进行对比，可以通过选择某些方程来显示其他拟合结果，如图 3-27 所示。



图 3-27 通过选择某些方程来显示其他拟合结果

单击"Save to workspace"按钮，打开"Save Fit to Workspace"对话框，如图 3-28 所示。进行设置，可以将拟合结果保存到 MATLAB 工作空间中。



图 3-28　"Save to workspace" 对话框

拟合结构为

```
fit1
fit1 =
      type: 'polynomial degree 3'
     coeff: [3.8555e-006 -0.0153 17.7815 -4.8519e+003]
```

再次选择 按钮，可以指定一个包含 $x$ 值的向量，计算这些位置上的拟合值。将该向量输入到 "Evaluate" 按钮后面的文本框中，然后单击 "Evaluate" 按钮。例如，如果输入向量 2000:10:2050，2000 年到 2050 年的人口数按照 10 年的增量进行估计。$x$ 值和对应值 $f(x)$ 显示在 "Evaluate" 下面的面板中，如图 3-29 中所示。

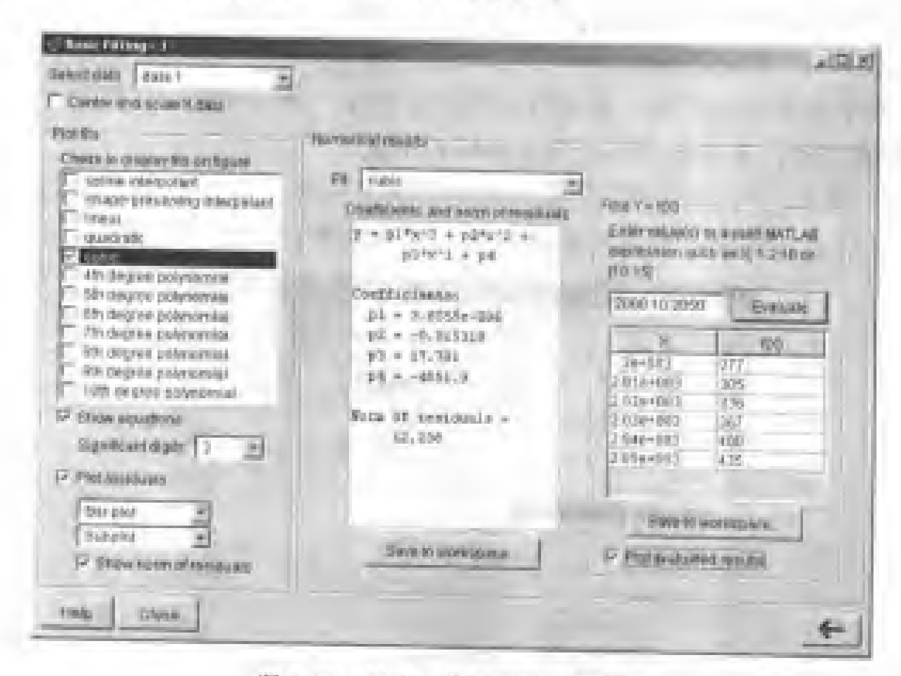


图 3-29　显示 x 值和对应的拟合值

选择 "Plot evaluated results" 核选框，显示当前图形中沿当前数据的计算值，如图 3-30 中所示。

可以通过选择 "Save to workspace" 按钮打开 "Save Results to Workspace" 对话框，如图 3-31 中所示。利用该对话框将计算值保存到 MATLAB 工作空间中。

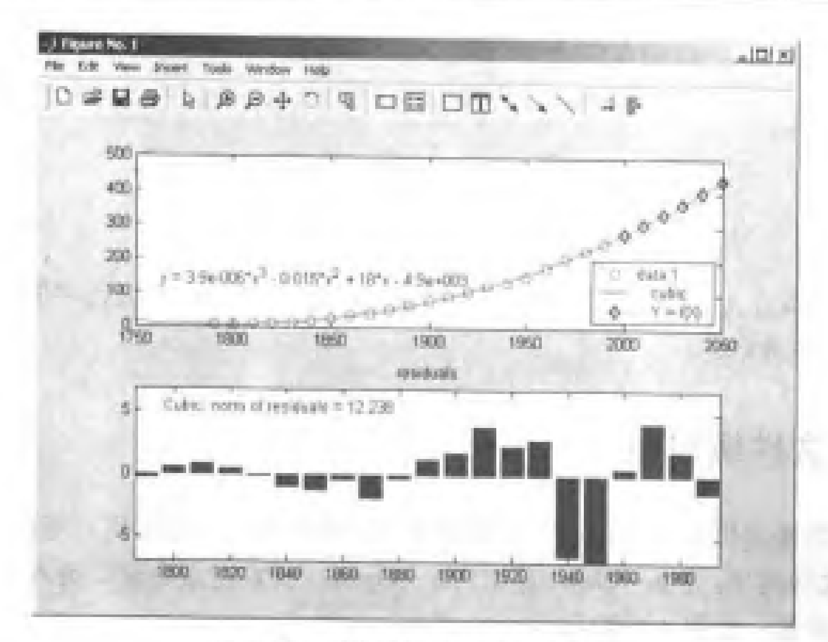



图 3-30 拟合结果和对应的残差图

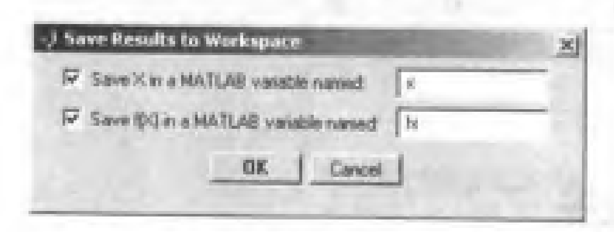



图 3-31 "Save Results to Workspace" 对话框

# 第 4 章　M 文件设计

M 文件是 MATLAB 中功能语句的集合。使用 M 文件，可以以程序的形式重复处理数据，从而提高工作效率。

## 4.1　M 文件编辑器

M 文件编辑器提供了一个进行文本编辑和 M 文件调试的图形用户界面，在 MATLAB 主界面中依次选择菜单：File→New 或者 File→Open，或者在命令窗口键入 edit 函数，可以打开 M 文件编辑器，如图 4-1 所示。

图 4-1　M 文件编辑器

## 4.2　脚本式 M 文件和函数式 M 文件

假设我们想绘一个单位球面，并且要让球面看起来比较平滑，可以在命令窗口中键入下面 3 行命令，其中，第 1 行是一个 sphere 命令函数，表示绘制一个单位球面；第 2 行命令对球面进行插值着色；第 3 行设置坐标系，使各坐标方向上的度量单位相同，如果不同，球面看起来会像椭球面。

```
sphere
shading interp
axis equal
```

生成图4-2所示的球面。

现在假设除了需要生成球面以外，还想生成柱面。可以考虑编制一个绘图函数，这个函数有一个surface参数。调用这个函数时，如果将该参数设置为“sphere”，生成一个单位球面；设置为“cylinder”时，生成一个单位柱面。

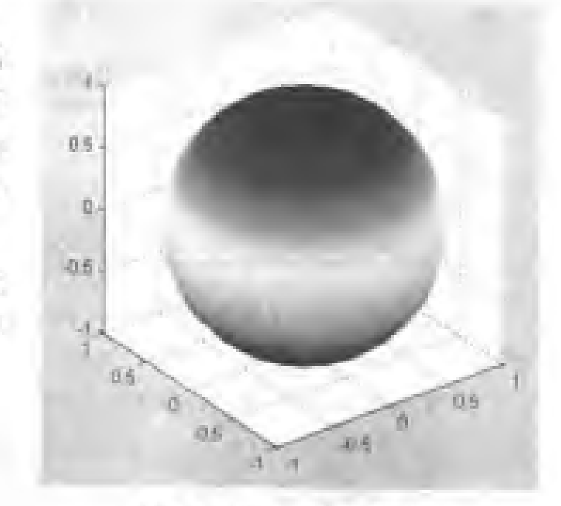


图4-2　生成一个单位球面

创建和编辑程序的工作在M文件编辑器中完成，在主界面中依次选择File→New→M-file菜单项，打开M文件编辑器。

在M文件编辑器中输入下面的代码。

```
function drawsur(surface)
    switch surface
      case 'sphere'
        sphere
      case 'cylinder'
        cylinder
    end
    shading interp
    axis equal
```

然后将它保存到MATLAB安装目录下的work目录下，名为drawsur.m。现在，可以在命令窗口中调用drawsur函数了。注意，如果保存在其他目录下，需要用当前目录窗口将该目录设置为当前目录。在命令窗口中键入

```
drawsur('sphere')
```

生成与图4-2中相同的球面。键入

```
drawsur('cylinder')
```

生成单位柱面，如图4-3所示。

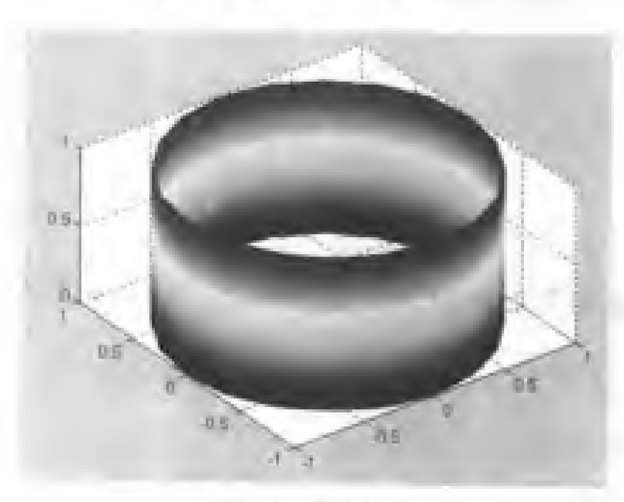


图4-3　单位柱面

M文件有两种，一种是脚本式M文件，另一种是函数式M文件。上面使用的是函数式形式。为了进行比较，我们继续使用前面的例子。新建一个M文件，在编辑器中输入

```
sphere
shading interp
axis equal
```

保存为spher.m。在命令窗口中键入

```
spher
```

生成单位球面。

很明显，这里的使用方式与drawsur的使用方式完全不同。它没有function关键字，没有输入变量。这就是脚本式M文件的使用方式，而drawsur文件中使用的是函数式M文件方式。

概括而言，脚本式 M 文件和函数式 M 文件的区别可归纳为表 4-1。

**表 4-1　脚本式 M 文件和函数式 M 文件的区别**

| 脚本式 M 文件 | 函数式 M 文件 |
|---|---|
| 不接受输入变量，没有返回值 | 可以接受输入变量，可以有返回值 |
| 基于工作空间中的数据进行操作 | 默认时，文件中变量的作用范围只限于函数内部 |
| 自动完成需要花费很多时间的多步操作时使用 | 扩展 MATLAB 语言功能时使用 |

函数式 M 文件比较标准的格式如下所示，其中黑体加粗的文字表示 M 文件的基本组成部分。

```
function [x, y] = myfun(a, b, c)        函数定义行
% H1 行 – 用一行文字来综述函数的功能
% 帮助文本 – 用一行或多行文本解释如何使用函数
% 在命令行中键入"help <functionname>"时可以使用它

% 函数体一般从第一个空白行后开始
% 注释 – 描述函数的行为，输入输出的类型等
% 在命令行中键入"help <functionname>"时不会显示这些文本

x = prod(a, b);                        % 开始编写函数代码
```

所以，一个完整的函数式 M 文件应该包括函数定义行、H1 行、帮助文本、函数体、注释和函数代码等项目。

## 4.3 流控制

MATLAB 中主要有 8 种流控制语句，包括 if, switch, while, for, continue, break, try...catch 和 return。下面分别进行介绍。

### 1．if/elseif 语句

最简单的 if/elseif 语句,如下：

```
if expression
    statements;
end
```

如果表达式中的值为真，就执行 if 与 end 语句之间的代码，否则跳过。如果有两个选项，使用下面的 if/elseif 语句。

```
if expression
    statements;
else
    statements;
end
```

有多个选择时可以类似地使用 if/elseif 语句。

### 2．switch/case 语句

该语句的通用格式为

```
switch switch-expression
case case-expression1,
       statements1;
case case-expresson2,
       statements2;
case case-expression3,
       statements3;
.....
otherwise
       statements;
end
```

其中，switch-expression 给出开关条件，当有 case-expression 与之匹配时，就执行其后的语句，如果没有 case-expression 与之匹配，就执行 otherwise 后面的语句。在执行过程中，只有一个 case 命令被执行，当执行完命令后，程序就跳出分支结构，执行 end 下面的语句。

表 4-2 中列出了 switch/case 语句与 if/elseif 语句之间的区别。

表 4-2　switch/case 语句和 if/elseif 语句之间的区别

| switch/case 语句 | if/elseif 语句 |
| --- | --- |
| 更好读 | 更难读 |
| 可以比较不同长度的字符串 | 需要 strcmp 函数比较不同长度的字符串 |
| 只测试相等性 | 测试相等性或不等性 |

### 3．while 循环

while 循环以不定的次数来求一组命令的值。while 循环的一般形式为

```
while expression
       statements;
end
```

只要表达式 expression 中的元素为真，就执行 while 和 end 语句之间的命令。

### 4．for 循环

for 循环允许一组命令以固定的和预定的次数重复执行。For 循环的一般形式为

```
for v=expression
       statements;
end
```

### 5．continue 命令

continue 命令经常与 for 或 while 语句一起使用，其作用是结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行下一次是否执行循环的判断。

### 6．break 命令

break 命令也经常与 for 或 while 等语句一起使用，其作用是终止本次循环，跳出最内层

的循环。使用 break 命令可以不必等到循环的自然结束，而是根据条件，退出循环。

### 7. return 命令

return 命令能使当前正在运行的函数正常退出，并返回调用它的函数，继续运行。这个语句经常用于函数的末尾，以正常结束函数的运行。

break,continue 和 return 函数比较容易混淆，表 4-3 中对它们进行了比较详细的比较。

表 4-3 break,continue 和 return 函数的区别

| 函 数 | 用 在 何 处 | 描 述 |
|---|---|---|
| break | for 或 while 循环 | 它出现时，退出循环，在嵌套循环中，进入相邻的外层循环 |
| continue | for 或 while 循环 | 在本循环中跳过剩余的语句，进入本循环的下一次迭代 |
| return | 任意位置 | 它出现时，立即退出函数，进入函数的调用函数中 |

### 8. try ... catch 语句

try…catch 语句进行错误捕获，它把有可能引起异常的语句放在 try 控制块中，这样当 try 控制块中 statement 语句引起异常时，catch 控制块就可以捕获它，并针对不同的错误类型，进行不同的处理。try…catch 命令的调用格式为

```
try,
    statement,
    ...,
    statement,
catch,
    statement,
    ...,
    statement,
end
```

## 4.4 函数变量

调用函数时，调用者通过一个变量列表传递数据，并获取返回值。MATLAB 采用的是传值方式。

### 4.4.1 检查输入变量的个数

利用 nargin 和 nargout 函数可以确定函数输入变量和输出变量的个数。然后可以根据变量个数用条件语句完成不同的任务，例如，

```
function c = testarg1(a, b)
if (nargin == 1)
    c = a .^ 2;
elseif (nargin == 2)
    c = a + b;
end
```

给定一个输入变量时，函数计算输入值的平方；给定两个输入变量时，求它们的和。

下面是一个更高级的示例，它寻找字符串中的第一个连续字符集。给定一个输入时，函数假设默认的间隔符为空格；给定两个输入时，会让你指定另外一个分隔符。还可以有两个可能的输出变量列表。

```
function [token, remainder] = strtok(string, delimiters)
% 至少需要一个输入变量的函数
if nargin < 1
    error('Not enough input arguments.');
end
token = []; remainder = [];
len = length(string);
if len == 0
    return
end
% 如果有一个输入变量，使用空格作为分隔符
if (nargin == 1)
    delimiters = [9:13 32]; % 空格字符
end
i = 1;
% 确定非分隔符字符开始的地方
while (any(string(i) == delimiters))
    i = i + 1;
    if (i > len), return, end
end
% 查找连续字符结束的地方
start = i;
while (~any(string(i) == delimiters))
    i = i + 1;
    if (i > len), break, end
end
finish = i - 1;
token = string(start:finish);
% 对于两个输出变量的情况，计算第一个分隔符后面的字符数
if (nargout == 2)
    remainder = string(finish+1:end);
end
```

strtok 函数是一个 MATLAB M 文件，位于 strfun 目录下。

### 4.4.2 传递变量

使用 varargin 和 varargout 函数可以传递任意个数的输入变量或者返回任意个数的输出给函数。

MATLAB 把所有指定的输入变量指定到一个单元数组中。每个单元可以包含任意大小或类型的数据。对于输出变量，函数代码必须把它们打包到一个单元数组中，这样，MATLAB 就可以把变量返回给调用函数。

下例中的函数接受任意个数的二元素向量，并用直线段连接它们

```
function testvar(varargin)
for k = 1:length(varargin)
    x(k) = varargin{k}(1); % 单元数组索引
    y(k) = varargin{k}(2);
end
xmin = min(0,min(x));
ymin = min(0,min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

按照这种方法编码，testvar 函数在使用时可以有不同的输入列表，例如，

```
testvar([2 3],[1 5],[4 8],[6 5],[4 2],[2 3])
testvar([-1 0],[3 -5],[4 2],[1 1])
```

### 4.4.3 解包 varargin 中的内容

因为 varargin 将所有输入变量包含在一个单元数组中，所以有必要使用单元数组索引来提取数据。例如，

```
y(n) = varargin{n}(2);
```

其中，索引表达式{n}获取 varargin 的第 $n$ 个单元。表达式(2)表示单元内容的第二个元素。

### 4.4.4 打包 varargout

当允许有任意多个输出变量时，必须将所有输出打包到 varargout 单元数组中。使用 nargout 确定输出变量的个数。例如，下面的代码接受两列输入数组，其中第一列表示一系列 $x$ 坐标变量，第二列表示 $y$ 坐标变量。它将数组分乘单独的[xi,yi]向量。可以将该向量传递给 testvar 函数。

```
function [varargout] = testvar2(arrayin)
for k = 1:nargout
    varargout{k} = arrayin(k,:);     % 单元数组赋值
end
```

for 循环中的赋值语句使用了单元数组的赋值语法。语句左侧的单元数组通过大括号进行索引。调用 testvar2 函数时，键入

```
a = [1 2; 3 4; 5 6; 7 8; 9 0];
[p1, p2, p3, p4, p5] = testvar2(a)
p1 =
     1     2
p2 =
     3     4
p3 =
     5     6
p4 =
     7     8
p5 =
     9     0
```

### 4.4.5 变量列表中的 varargin 和 varargout

varargin 或 varargout 必须出现在变量列表的最后，即函数调用必须首先指定必要的变量。例如，下面的函数声明行显示了 varargin 和 varargout 的正确位置。

```
function [out1,out2] = example1(a,b,varargin)
function [i,j,varargout] = example2(x1,y1,x2,y2,flag)
```

### 4.4.6 返回输出变量

放在函数定义行中等号左侧的任何变量都是调用函数的返回值。如果传递任何函数可以修改的输入变量，需要把相同的变量作为输出变量，这样，调用函数可以获得更新后的值。例如，

```
[text, offset] = readText(filestart, offset)
```

## 4.5 子函数和私有函数

M 文件中可以包含一个以上的函数。文件中除主函数以外的其他函数称为子函数，并且它们只对主函数或同一文件中的其他子函数可见。

主函数放在最上面，子函数放在下面，并且各子函数不分先后顺序。

```
function [avg, med] = newstats(u) % 主函数
% 本函数用内部函数计算均值和中值
n = length(u);
avg = mean(u, n);
med = median(u, n);

function a = mean(v, n)         % 子函数
% 计算均值
a = sum(v)/n;
```

```
function m = median(v, n)          % 子函数
% 计算中值
w = sort(v);
if rem(n, 2) == 1
    m = w((n+1) / 2);
else
    m = (w(n/2) + w(n/2+1)) / 2;
end
```

子函数 mean 和 median 计算输入数据的均值和中值。主函数 newstats 确定变量列表的长度并调用子函数，将列表长度 $n$ 传递给它们。

即使在同一个 M 文件中，子函数也不能获取用于主函数或其他子函数的变量，除非将它们声明为全局变量，或者作为变量传递。

从 M 文件内部调用函数时，MATLAB 首先检查文件，看函数是否子函数。然后检查该名称的私有函数，最后搜索路径上的独立 M 文件或内部函数。因为首先检查子函数，所以可以用相同名称的子函数覆盖已经存在的 M 文件。

私有函数是 private 子目录中的函数。它们只对父目录中的函数可见。例如，假设目录 newmath 位于 MATLAB 搜索路径上，则 newmath 目录的 private 子目录中包含的函数只能被 newmath 目录中的函数调用。

因为私有函数在父目录外是不可见的，在其他目录中可以使用与之相同的名称。

## 4.6 编程技巧

### 4.6.1 函数句柄

可以给任何 MATLAB 函数创建一个句柄，然后把该句柄用作引用函数的一种方式。将函数句柄作为变量传递给其他函数以后，其他函数可以用这个句柄执行对应的函数。

在 MATLAB 中用符号@创建函数句柄，将该符号放在函数名的前面。下面的例子为 sin 函数创建一个函数句柄，并将它赋给变量 fhandle。

```
fhandle = @sin;
```

可以用该句柄调用函数，调用方式与使用函数名时的一样，即

```
fhandle(arg1, arg2, ...);
```

下面的函数 plot_fhandle 接收函数句柄和数据，用函数句柄生成 $y$ 轴数据，并利用该数据绘图。

```
function x = plot_fhandle(fhandle, data)
plot(data, fhandle(data))
```

将 sin 函数的句柄和下面的变量作为参数调用 plot_fhandle 函数时，生成正弦波图形。

```
plot_fhandle(@sin, -pi:0.01:pi)
```

### 4.6.2　函数的函数

有一类函数称为函数的函数，即它们在其他函数的基础上进行工作。函数的函数主要用在方程求解、最优化、积分和一般差分方程求解等方面。

MATLAB 用函数式 M 文件表示非线性函数。例如，下面是 MATLAB 安装目录下 demos 目录中 humps 函数的简化版本。

```
function y = humps(x)
y = 1./((x-.3).^2 + .01) + 1./((x-.9).^2 + .04) - 6;
```

计算该函数在 0 到 1 之间一系列点上的值：

```
x = 0:.002:1;
y = humps(x);
```

然后绘图：

```
plot(x,y)
```

生成图 4-4。

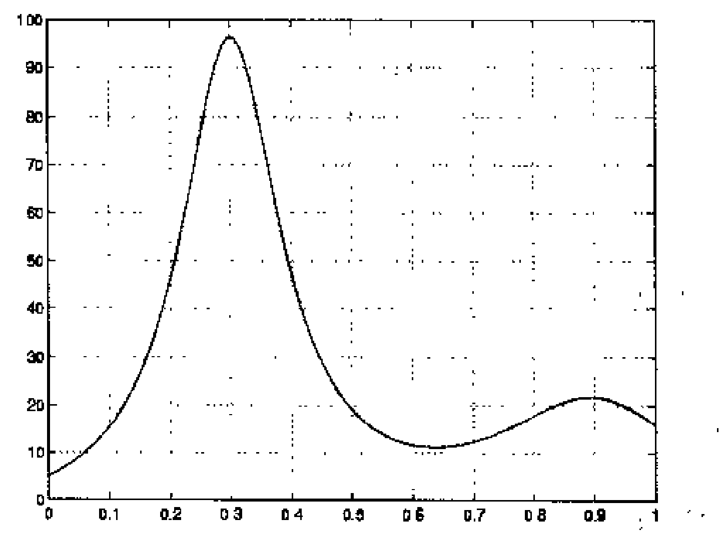



图 4-4　生成数据的图形

图形显示该函数在 x=0.6 附近有一个局部极小点。用 fminsearch 函数可以查找极小点。下面将 humps 函数的句柄和极小点的位置初值作为参数调用 fminsearch 函数。

```
p = fminsearch(@humps,.5)
p =
    0.6370
```

下面的语句计算极小点处的函数值。

```
humps(p)
ans =
    11.2528
```

### 4.6.3 向量化

使程序运行更快的方法之一是将构造程序的算法进行向量化。在其他程序语言可能使用 for 循环或 do 循环的地方，MATLAB 可以使用向量或矩阵操作。对于下面的程序

```
x = .01;
for k = 1:1001
    y(k) = log10(x);
    x = x + .01;
end
```

进行向量化以后，得

```
x = .01:.01:10;
y = log10(x);
```

但是，对于复杂的代码，向量化的效果并不明显。

### 4.6.4 预分配内存空间

如果不能向量化某段代码，可以通过预分配保存输出的任何向量或数组的内存空间来加速 for 循环。例如，下面的代码用函数 zeros 预分配 for 循环中创建的向量的内存空间，使得这个 for 循环的运行速度显著加快。

```
r = zeros(32,1);
for n = 1:32
    r(n) = rank(magic(n));
end
```

前面的例子没有使用内存预分配，每循环一次，MATLAB 解释器就会将 *r* 向量的元素增加一个。而内存预分配以后，就取消了这个步骤，从而使运行加速。

## 4.7 面向对象编程

在 MATLAB 中可以实现面向对象编程。包括使用类和对象，创建构造函数、进行函数和运算符重载和使用继承等。使用设计良好的类时，面向对象编程可以显著提高代码的重用性。并使程序更易于维护和扩展。

设计一个 MATLAB 类时，应该提供一个使类能够在 MATLAB 环境中稳定运行的标准的方法集合。利用这些方法，应该可以实现创建类、设置属性以及对对象的引用、赋值、索引和转换等操作。表 4-4 列出了 MATLAB 类中包括的基本方法。

表 4-4　MATLAB 类中的基本方法

| 方　法 | 描　述 |
|---|---|
| class constructor | 创建类的构造函数 |
| display | MATLAB 是否显示对象的内容 |
| set 和 get | 设置和获取类属性 |
| subsref 和 srbsasgn | 用户对象的索引引用和赋值 |

续表

| 方　法 | 描　述 |
|---|---|
| end | 在使用了对象的索引表达式中支持 end 语法，如 A(1:end) |
| subsindex | 支持在索引表达式中使用对象 |
| converters(如 donble.char) | 将对象转换为一种 MATLAB 数据类型 |

下面的例子创建一个 polynom 类。该类为数据保存指定一个结构，并定义操作 polynom 对象的方法路径(@polynom)。

**1．polynom 类的数据结构**

polynom 类表示一个带有行向量的多项式，行向量中包含降序排列的变量幂次项的系数。这样，polynom 对象 *p* 是一个具有单一字段 *p, c* 的结构。*p, c* 中包含各系数。本字段只在@polynom 目录中的方法内可以获得。

**2．polynom 类的方法**

为了创建一个在 MATLAB 环境中运行良好的类，并为多项式数据类型提供有用的功能，polynom 类实现下面的方法。

- 构造函数 polynom.m;
- double 转换;
- char 转换;
- display 方法;
- subsref 方法;
- 重载的 +, - 和 * 操作符;
- 重载的 roots.polyval.plot 和 diff 函数。

（1）polynom 类的构造函数

下面是 polynom 类的构造函数@polynom/polynom.m

```
function p = polynom(a)
if nargin == 0
    p.c = [];
    p = class(p,'polynom');
elseif isa(a,'polynom')
    p = a;
else
    p.c = a(:).';
    p = class(p,'polynom');
end
```

polynom 类有 3 个构造函数，它们具有不同的形式。

- 没有输入变量　如果调用没有变量的构造函数，它返回一个带空字段的 polynom 对象。
- 输入变量为对象　如果调用输入变量为对象的构造函数，MATLAB 返回该输入变量。
- 输入变量为系数向量　如果输入变量是一个变量，该变量不是 polynom 对象，则将它改写为行向量的形式，并将它指定为对象结构的字段，class 函数创建 polynom 对象，并在后面由构造函数返回。下面是使用 polynom 构造函数的一个例子。

```
p=polynom(1 0 -2 -5)
```

这将创建一个指定系数的多项式。

（2）转换器方法

转换器方法将一个类的对象转换为另一个类的对象。MATLAB 类中最重要的转换器方法中有两个是 double 和 char。转换为 double 将生成 MATLAB 传统矩阵，尽管它对某些类可能不合适。转换为 char 对于生成打印输出很有用。

① polynom 到 Double 的转换

polynom 类的 double 转换方法是一个简单的 M 文件：@polynom/double.m，它只提取对象 $p$ 的系数向量 p=polynom(1　0　-2　-5)

```
function c = double(p)
    c = p.c;
```

语句 double(p)返回

```
ans=
    1    0    -2    -5
```

② polynom 到 char 的转换

char 转换是一个关键的方法，因为它生成一个与独立变量 $x$ 的幂次有关的字符串。所以，一旦已经指定了 $x$，返回的字符串是一个语法上正确的 MATLAB 表达式。这里为 @polynom/char.m。

```
function s = char(p)
    if all(p.c == 0)
        s = '0';
    else
        d = length(p.c) - 1;
        s = [];
        for a = p.c;
            if a ~= 0;
                if ~isempty(s)
                    if a > 0
                        s = [s ' + '];
                    else
                        s = [s ' - '];
                        a = -a;
                    end
                end
                if a ~= 1 | d == 0
                    s = [s num2str(a)];
                    if d > 0
                        s = [s '*'];
                    end
                end
                if d >= 2
```

```
                s = [s 'x^' int2str(d)];
            elseif d == 1
                s = [s 'x'];
            end
        end
        d = d - 1;
    end
end
```

如果创建 polynom 对象 *p*(p=polynom(1 0 -2 -5);)，然后对 *P* 调用 char 方法，char(p),则 MATLAB 生成结果。

```
ans =
    x^3 - 2*x - 5
```

char 返回的值是一个字符串，一旦为 *x* 定义了标量值，可以将它传递给 eval 函数。如，

```
x = 3;
eval(char(p))
ans =
    16
```

（3）polynom 的 display 方法

其 M 文件为@polynom/display.m 本方法依赖于 char 方法，生成一个表示多项式的字符串，然后，它显示在屏幕上，本法生成的输出与 MATLAB 的标准输出相同。即，变量名后面跟等号，然后空一行，然后在一个新行中显示值。:

```
function display(p)
disp(' ');
disp([inputname(1),' = '])
disp(' ');
disp(['     ' char(p)])
disp(' ');
```

语句 p=polynom(1 0 -2 -5)创建一个 polynom 对象，因为语句不以分号结束，结果输出为

```
p =
    x^3 - 2*x - 5
```

（4）polynom 的 subsref 方法

对于 polynom 对象 p

```
p = polynom([1 0 -2 -5]);
```

下面的子脚本表达式返回 *x*=3 和 *x*=4 处多项式的值：*p*([3 4])

```
ans =
    16    51
```

subsref 方法利用 polynom 类中定义的 char 方法，生成一个可以计算的表达式。

```
function b = subsref(a,s)
    switch s.type
    case '()'
```

```
            ind = s.subs{:};
            for k = 1:length(ind)
                b(k) = eval(strrep(char(a),'x',num2str(ind(k))));
            end
        otherwise
          error('Specify value for x as p(x)')
        end
```

一旦多项式表达式已经由 char 方法生成，使用 strrep 函数交换字符 *x*，以传递值的内容。然后 eval 函数评价表达式并在输出变量中返回值。

（5）重载算术操作符

有些算术操作符对于多项式是有意义的，应该在 polynom 类中实现，重载算术操作符时，记住进行操作的数据类型。本例中，为 polynom 类定义了 plus、minus 和 mtimes 方法，以便实现加、减和乘等运算。

① 重载 + 操作符

如果 *p* 或 *q* 为 polynom 类对象，表达式 *p+q* 调用@polynom/plas.m 函数，下面的 M 文件重新为 polynom 类定义 + 操作符。

```
function r = plus(p,q)
p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] + [zeros(1,-k) q.c]);
```

该函数首先将两个输入变量定义为多项式。这样保证至少有一个输入不是多项式的情况下函数也能正常进行。然后函数获取两个系数向量。并且在必要时将其中的某个置 0，以使它们具有相同的长度。实际求和是求两个多项式的系数的和。用向量表示，最后，函数第三次调用 polynom 构造函数，得到结果。

② 载操作符 - (减号)

可以用相同的方法实现 - 操作符。MATLAB 调用@polynom/minus.m 计算 *p-q*。

```
function r = minus(p,q)
p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] - [zeros(1,-k) q.c]);
```

③ 重载 * 操作符

MATLAB 调用方法@polynom/mtimes.m 计算结果 *p*q*。因为它从 MATLAB 矩阵乘法(matrix multiplication)中继承而来，所以函数名的第一个字母为 *m*。

```
function r = mtimes(p,q)
    p = polynom(p);
    q = polynom(q);
    r = polynom(conv(p.c,q.c));
```

④ 使用重载的操作符

给定 polynom 对象 p = polynom([1 0 -2 -5])，MATLAB 在使用语句 $q = p+1$ 时调用函数 @polynom/plus.m 和@polynom/mtimes.m。

```
q = p+1
r = p*q
```

生成

```
q =
    x^3 - 2*x - 4
r =
    x^6 - 4*x^4 - 9*x^3 + 4*x^2 + 18*x + 20
```

（6）重载函数

MATLAB 已经有几个可以操作多项式的函数，它们可以通过重载，用在新的 polynom 对象中。在许多情况下，重载方法可以将原始函数应用于系数字段。

① 为 polynom 类重载 roots 方法

方法@polynom/roots.m 查找 polynom 对象的根。

```
function r = roots(p)
    r = roots(p.c);
    The statement roots(p)
    results in ans =
        2.0946
        -1.0473 + 1.1359i
        -1.0473 - 1.1359i
```

② 为 polynom 类重载 polyval 方法

polyval 方法计算给定点集的多项式。@polynom/polyval.m 使用嵌套的乘法或 Horner 法减少计算 $x$ 的不同幂次的乘法操作次数。

```
function y = polyval(p,x)
y = 0;
for a = p.c
    y = y.*x + a;
end
```

③ 为 polynom 类重载 plot 方法

用 root 和 polyval 重载 plot 方法。

```
function plot(p)
    r = max(abs(roots(p)));
    x = (-1.1:0.01:1.1)*r;
    y = polyval(p,x);
    plot(x,y);
    title(char(p))
    grid on
```

④ 重载 diff 方法

方法@polynom/diff.m 对一个多项式进行求导。方法是幂次减 1，用原来的幂次乘上对应的系数。

```
function q = diff(p)
c = p.c;
d = length(c) - 1;
q = polynom(p.c(1:d).*(d:-1:1));
```

下面用 methods 命令列出类的所有方法。

```
methods polynom
Methods for class polynom:
chardisplayminusplotpolynomrootsdiffdoublemtimespluspolyvalsubsref
```

下面先对两个 polynom 对象 $x$ 和 $p$ 作运算，然后绘图。

```
x = polynom([1 0]);
p = polynom([1 0 -2 -5]);
plot(diff(p*p + 10*p + 20*x) - 20)
```

# 第 5 章　图形用户界面（GUI）设计

GUI 是实现人机交互的中介，可以通过它实现数据输入、处理和输出。MATLAB 提供了一个专门的 GUI 设计工具——GUIDE。使用该工具，可以快速完成 GUI 设计任务。

## 5.1　GUIDE 简介

MATLAB 图形用户界面开发环境（GUIDE）提供了一系列创建图形用户界面（GUI）的工具。这些工具极大地简化了 GUI 设计和生成的过程。可以用 GUIDE 完成下面的任务。

- 输出 GUI：
- 使用 GUIDE 输出编辑器，可以通过单击和拖拉 GUI 组件很容易地创建 GUI；
- GUI 编程：
- GUIDE 自动生成一个控制 GUI 如何操作的 M 文件。该 M 文件初始化 GUI 并包含一个所有 GUI 回调（用户单击 GUI 组件时执行的命令）的框架。使用 M 文件编辑器，可以向回调中添加代码，运行相关函数。

### 5.1.1　启动 GUIDE

在命令窗口中键入 guide，启动 GUIDE，显示图 5-1 所示的“GUIDE Quick Start”对话框。

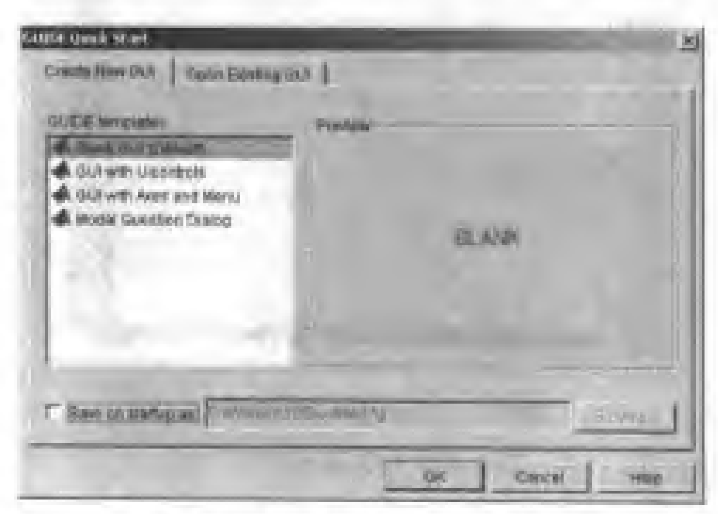



图 5-1　“GUIDE Quick Start”对话框

打开“GUIDE Quick Start”对话框，利用 GUIDE 模板创建新的 GUI，或者打开已经存在的 GUI。

选择这些选项中的一个以后，单击“OK”按钮，在输出编辑器中打开 GUI。

### 5.1.2　输出编辑器

在 GUIDE 中打开 GUI 以后，它显示在输出编辑器中。它是所有 GUIDE 工具的控制面板。图 5-2 显示了带有空白 GUI 模板的输出编辑器。

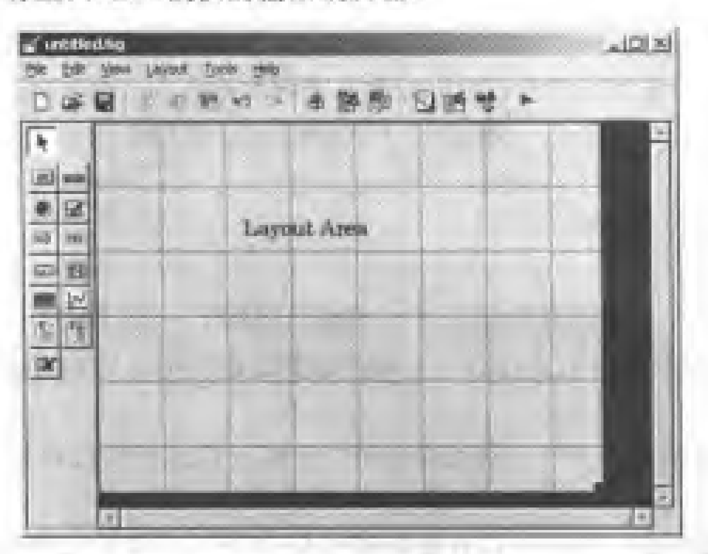



图 5-2　输出编辑器

可以通过拖拉组件输出 GUI，这些组件位于输出编辑器左侧的工具箱中，有按钮、弹出式菜单和坐标系等多种。例如，如果把一个按钮拖拉到输出区域，效果如图 5-3 所示。

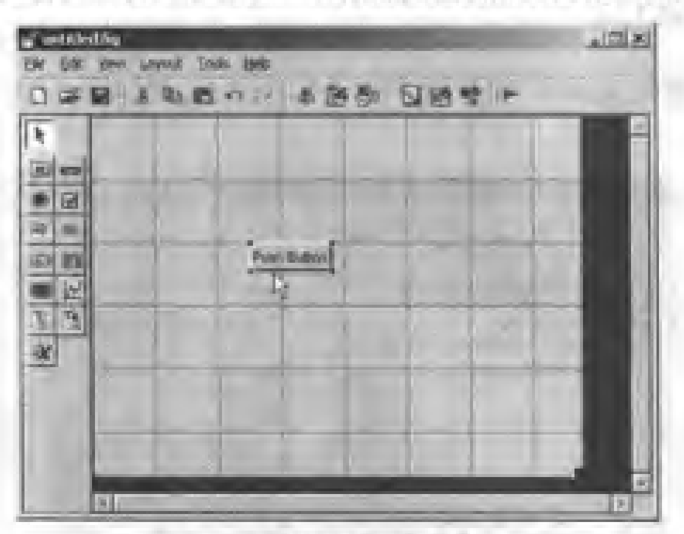


图 5-3　在输出编辑器中拖放组件

也可以用输出编辑器设置 GUI 组件的基本属性。

### 5.1.3　GUIDE 模板

"GUIDE Quick Start" 对话框提供了几种基本类型的 GUI 模板。使用模板的好处是可以通过快速修改模板来创建 GUI。选择一种模板以后，它的预览效果显示在右面的面板中。例如，选择 "GUI with Axes and Menu" 模板以后，"GUIDE Quick Start" 对话框如图 5-4 所示。

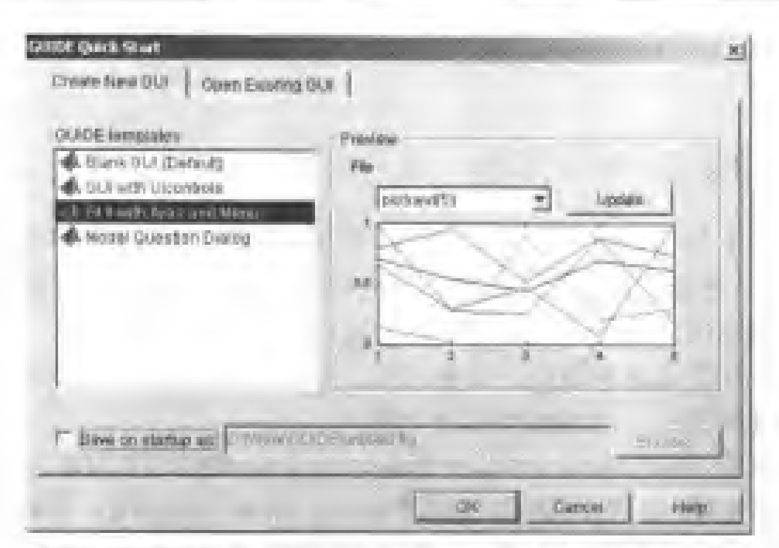


图5-4 选择“GUI with Axes and Menu”模板以后对话框的显示

单击“OK”按钮，在输出编辑器中打开面板，效果如图5-5所示。



图5-5 在输出编辑器中打开面板

要显示组件面板中GUI的名称，从“File”菜单中选择“Preferences”选项，选择“Show names in component palette”后面的核选框，单击“OK”按钮。

## 5.1.4 运行GUI

从“Tools”菜单中选择“Run”选项，或单击▶按钮，运行GUI。这将在输出编辑器外显示功能GUI。例如，运行有坐标系和菜单的GUI模板时，效果如图5-6所示。

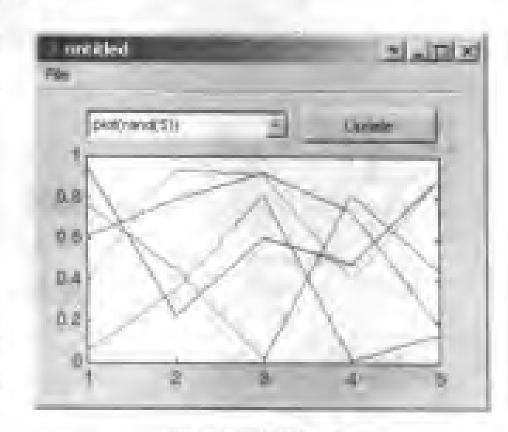


图5-6 运行界面

该 GUI 显示不同的 MATLAB 图形，从下拉式列表框中选择一个，然后单击“Update”按钮。

### 5.1.5 GUI FIG 文件和 M 文件

GUIDE 把 GUI 保存在两个文件中，它们在第一次保存或运行时生成。一个是 FIG 文件，扩展名为 .fig，它包含对 GUI 和 GUI 组件的完整描述；另一个是 M 文件，扩展名为 .m，它包含控制 GUI 的代码，包括其组件的回调。

这两个文件与 GUI 显示和编程任务相对应。在界面编辑器中创建 GUI 时，内容保存在 M 文件中；对 GUI 编程时，内容保存在 M 文件中。

设计好 GUI 以后，可以用 M 文件编辑器编写 GUI M 文件。GUIDE 会在第一次保存或运行 GUI 时生成这个文件。这个 GUI M 文件会初始化 GUI，并在 GUI 显示在屏幕上以前包含代码来完成任务，例如创建数据或图形。它包含每次用户单击 GUI 组件时运行的回调函数。

初始情况下，每个回调都只包含一个函数定义行，然后用 M 文件编辑器添加代码来完成函数的编写。单击输出编辑器工具条上的图标打开 M 文件。图 5-7 显示了“GUI with Axes and Menu”模板的 M 文件。

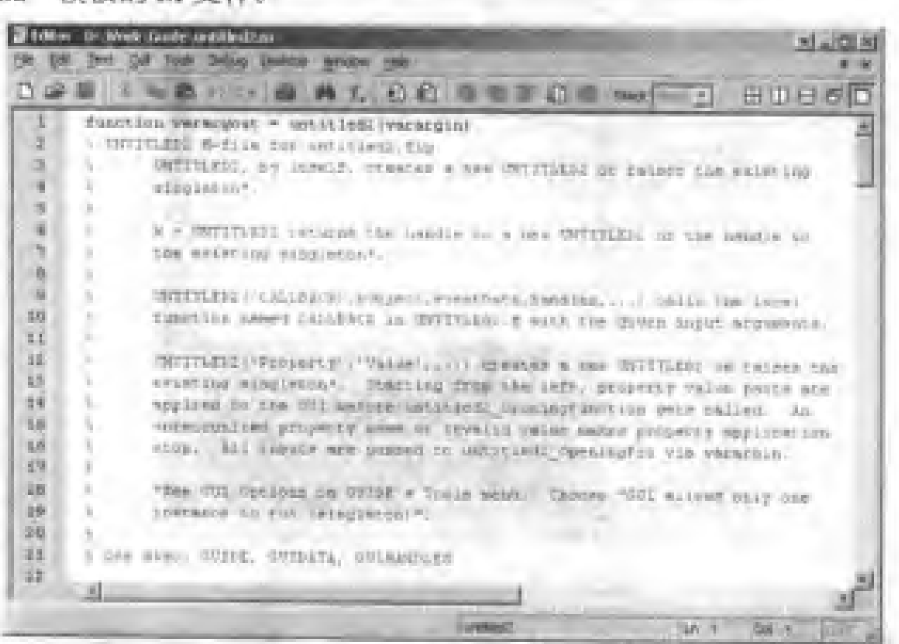


图 5-7 打开 M 文件

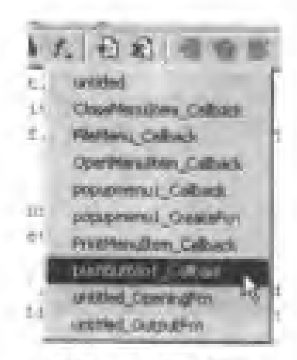



图 5-8 回调列表

可以通过在工具条中单击 按钮来查看任何 GUI 组件的回调。该操作显示所有回调的列表，如图 5-8 中所示。

在列表中单击回调可以显示 M 文件中包含回调的那一部分，可以编辑它。图 5-9 中显示了回调函数 pushbutton1_Callback。

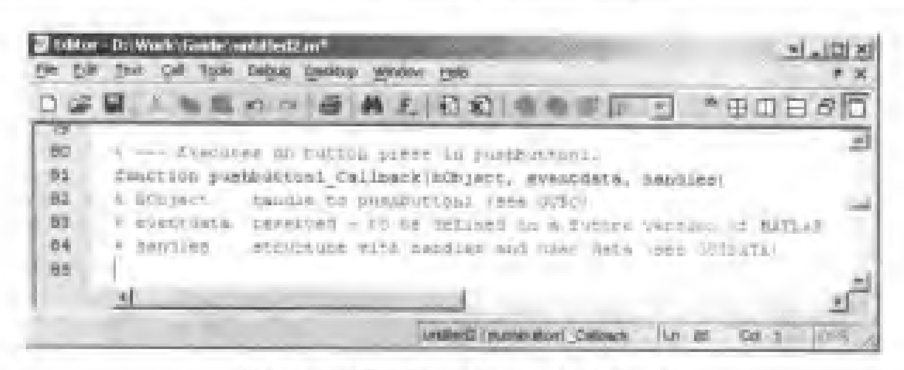


图 5-9　回调函数 pushbutton1_Callback

## 5.2　创建 GUI

下面结合一个例子来介绍用 GUIDE 创建 GUI 的过程。

### 5.2.1　设计 GUI

本例中用到的 GUI 包含一个坐标系，用于显示曲面、网格或等值线图，可以从下拉式列表框中选择绘图数据。图 5-10 显示了将要创建的界面的布局。

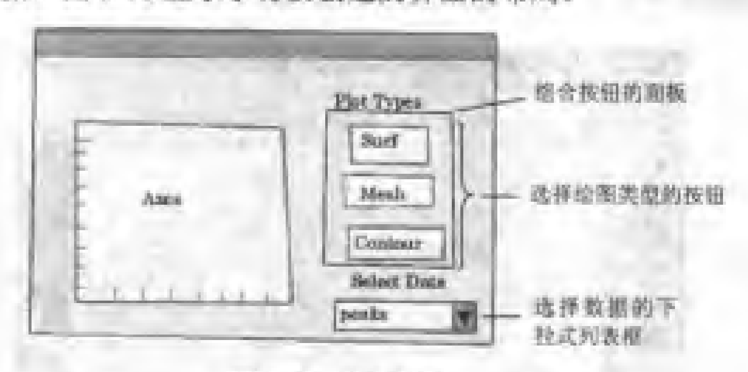



图 5-10　界面布局

面板中包含 3 个按钮，利用它们选择绘图类型。下拉式列表框中包含 3 个字符串，即 peaks, membrane 和 sine，它们提供了绘图数据。可以从菜单中选择数据并绘图。

### 5.2.2　完成 GUI

下面演示如何在 GUI 中输出组件。用 GUIDE 完成一个 GUI 的设计需要下面几个主要步骤：

- 在输出编辑器中打开一个新的 GUI 设计窗口；
- 设置 GUI 的大小；
- 在 GUI 上添加组件；
- 对齐组件。

**1．在输出编辑器中打开一个新的 GUI 设计窗口**

在 MATLAB 命令窗口键入 guide，打开 GUIDE，显示 “GUIDE Quick Start” 对话框，

如图 5-11 所示。



图 5-11　打开“GUIDE Quick Start”对话框

如果 GUIDE 已经打开，从“File”菜单中选择“New”选项，可以显示一个相似的对话框。这个对话框中没有“Open Existing GUI”选项卡。

在“GUIDE Quick Start”对话框中，选择“Blank GUI（Default）”模板，单击“OK”按钮，在输出编辑器中显示空的 GUI，如图 5-12 所示。

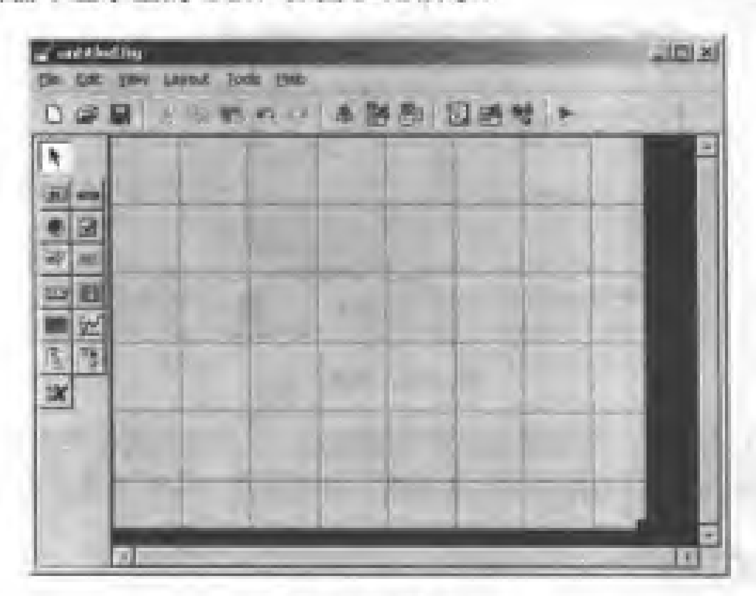


图 5-12　显示空的 GUI

要在组件面板中显示 GUI 组件的名称，从“File”菜单中选择“Preferences”选项，单击“Show names in component palette”后的核选框，单击“OK”按钮，现在输出编辑器如图 5-13 所示。

**2．设置 GUI 窗口的大小**

改变输出编辑器中网格区域的大小可以指定 GUI 的大小。单击网格区域的右下角并进行拖拉，可以改变网格区域的大小，如图 5-14 所示。

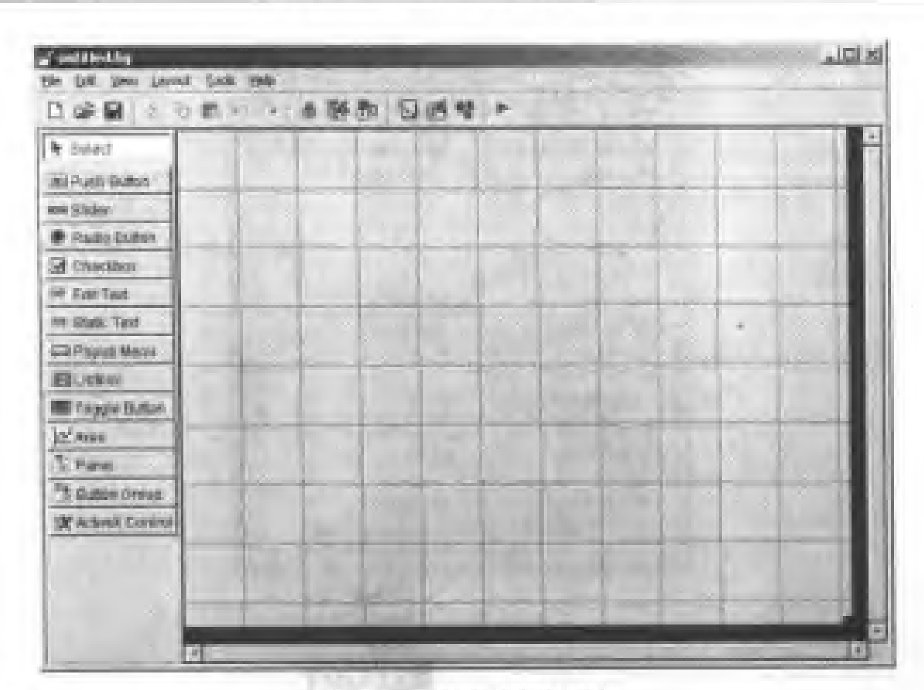


图 5-13 显示组件的名称

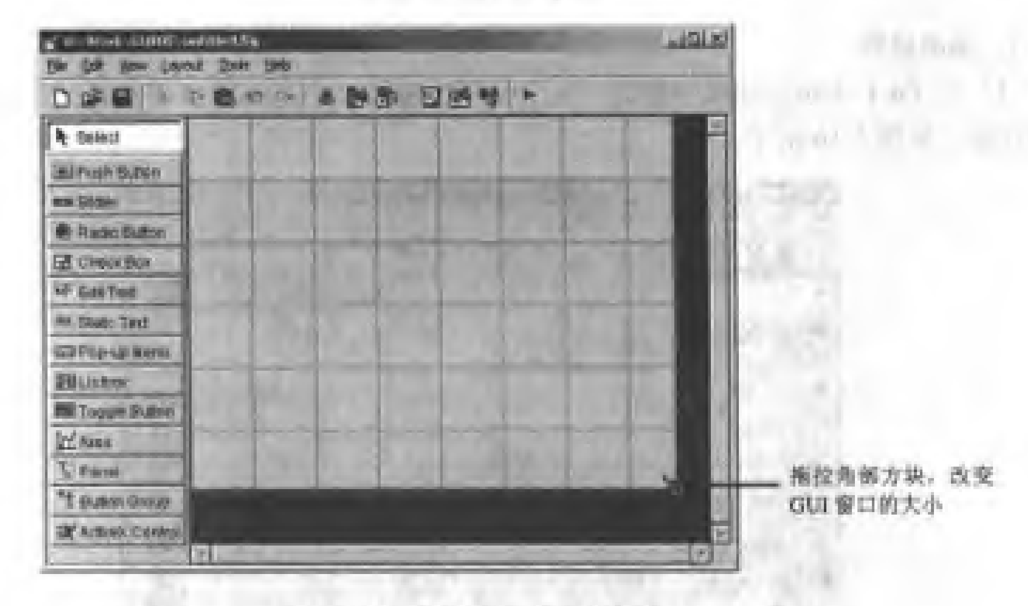



图 5-14 改变GUI窗口的大小

如果把GUI的位置和大小设置为精确值，完成下面的操作：

（1）从“View”菜单中选择“Property Inspector”选项，打开属性查看器，如图5-15所示；

（2）选择“Units”后面的按钮，然后从弹出式菜单中选择“inches”选项；

（3）单击“Position”后面的“+”符号；

（4）键入GUI左下角点的$x$和$y$坐标，以及GUI窗口的宽度和高度；

（5）将Units属性值重设为characters。

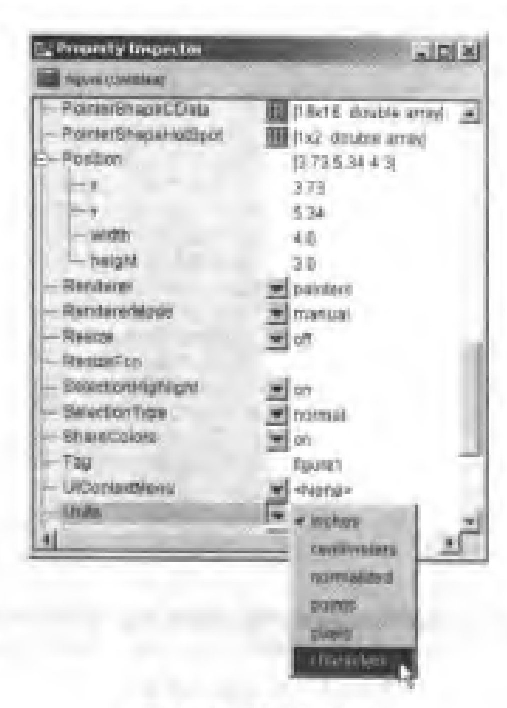


图 5-15　属性查看器

3．添加组件

（1）在 GUI 中添加面板和按钮。从工具箱中选择一个面板和 3 个按钮，并把它们拖拉到设计区，如图 5-16 所示。

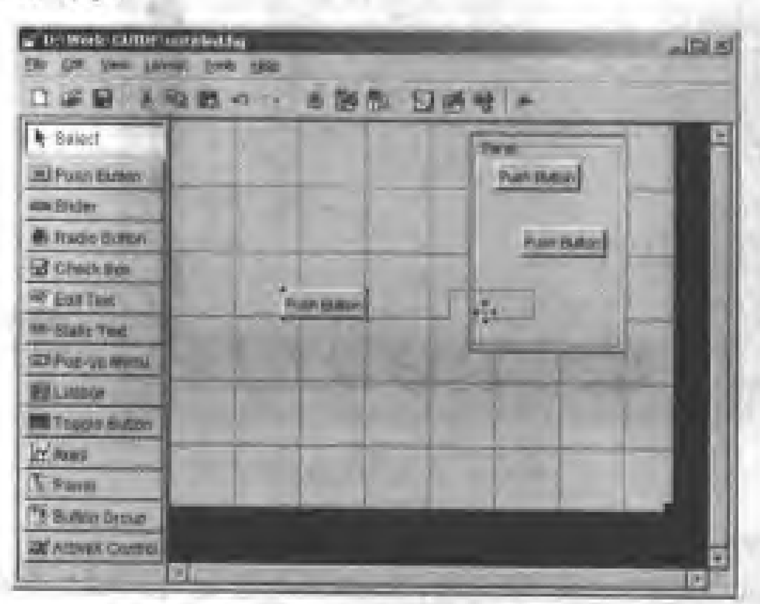


图 5-16　拖拉组件

（2）然后在 GUI 中添加一个文本框、一个下拉式列表框和一个坐标系。将各组件如图 5-17 所示进行布置。

4．对齐组件

如果组件具有相同的父对象，可以用对齐工具将各组件对齐。例如，下面对齐图 5-17 中的 3 个按钮：

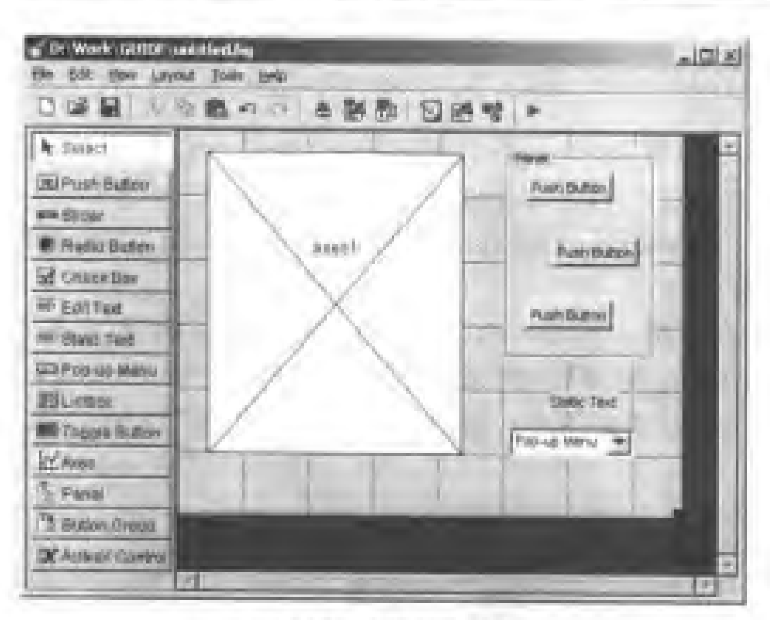


图 5-17　各组件的布置

（1）按住<Ctrl>键的同时连续单击这3个按钮，选择它们；

（2）从“Tools”菜单中选择“Align Objects”选项，显示对齐工具；

（3）如图5-18中所示，在对齐工具中作如下设置：

- 各按钮的垂向间隔为20像素；
- 各按钮在水平方向上左对齐。

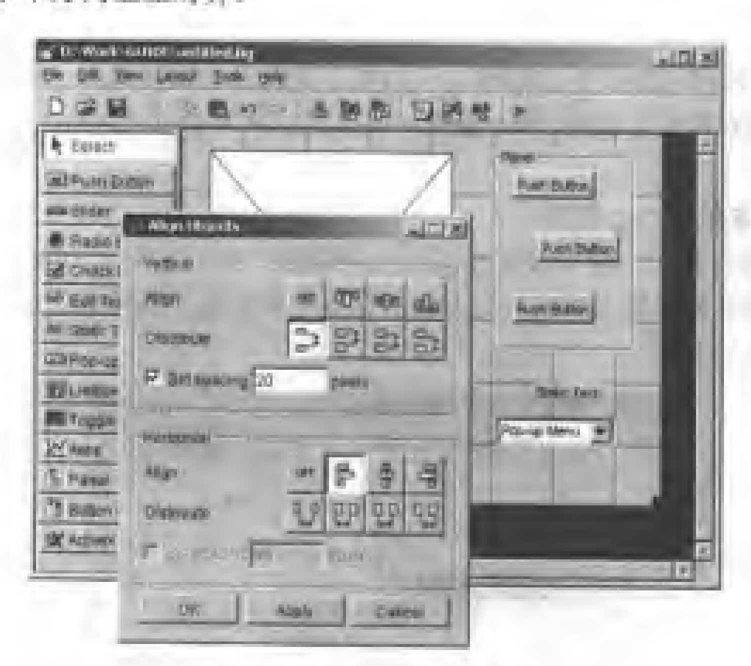


图 5-18　进行组件对齐设置

（4）单击“OK”按钮。

然后将坐标系和面板的顶部对齐。

### 5.2.3 设置 GUI 组件的属性

从“View”菜单中选择“Porperty Inspector”选项，显示“Property Inspector”对话框，利用该对话框设置每个 GUI 组件的属性。在输出编辑器中选择组件时，属性查看器会显示该组件的属性。如果没有选择组件，属性查看器会显示 GUI 图形窗口的属性。

**1. Name 属性**

图形窗口的 Name 属性值表示 GUI 的标题，显示在 GUI 的顶部。第一次保存或运行 GUI 时，GUIDE 会将 Name 属性的值设置为 FIG 文件的名称。一旦保存了 GUI，就可以将 Name 属性的值设置为标题字符串。如图 5-19 所示，将本 GUI 的 Name 属性值设置为 Simple GUI。

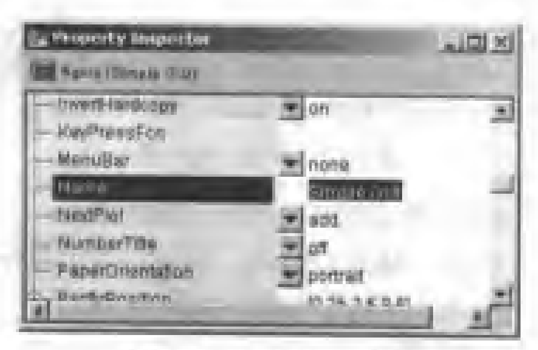



图 5-19　设置 Name 属性

**2. Title 属性**

面板的 Title 属性控制面板的标题，它出现在面板的顶部或底部。在输出编辑器中选择面板，然后在属性查看器中找到 Title 属性，将其值设置为 Plot Types。用 TitlePosition 属性控制标题的位置。

**3. 按钮和文本的 String 属性**

可以在某些用户界面控件中设置标签，例如按钮，利用它的 String 属性就可以进行设置。在输出编辑器中选择最上面的按钮，然后在属性查看器中找到 String 属性，将它的值设置为 Surf，如图 5-20 所示。

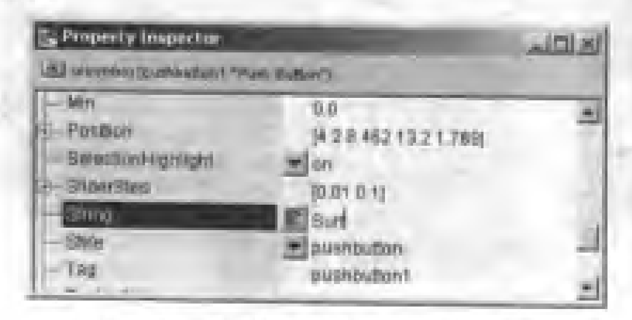



图 5-20　设置按钮和文本的 String 属性

可以通过单击输出编辑器查看组件的变化。类似地，将中间按钮的 String 属性值设置为 Mesh，将最下面按钮的 String 属性设置为 Contour，将文本的 String 属性设置为 Select Data。

#### 4．下拉式列表框的 String 属性

下拉式列表框的 String 属性控制下拉式列表框中的选项。在输出编辑器中选择下拉式列表框，可以设置下拉式列表框的选项。在属性查看器中，单击 String 属性后面的按钮，打开 String 属性的编辑窗口，如图 5-21 中所示。在 String 属性编辑窗口中删除“Pop-up”选项，并在 3 个单独的行中键入 peaks,membrane 和 sinc，如图 5-21 中所示。

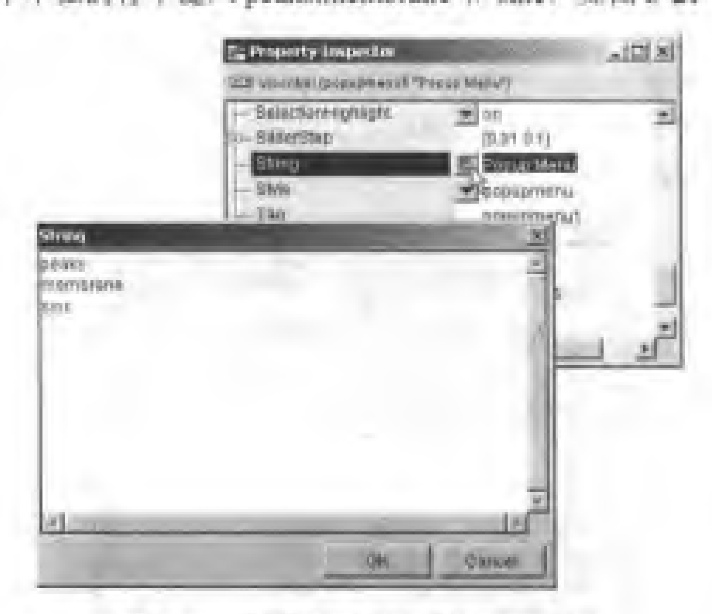



图 5-21 设置下拉式列表框的 String 属性

单击输出编辑器，GUI 的当前输出如图 5-22 所示。

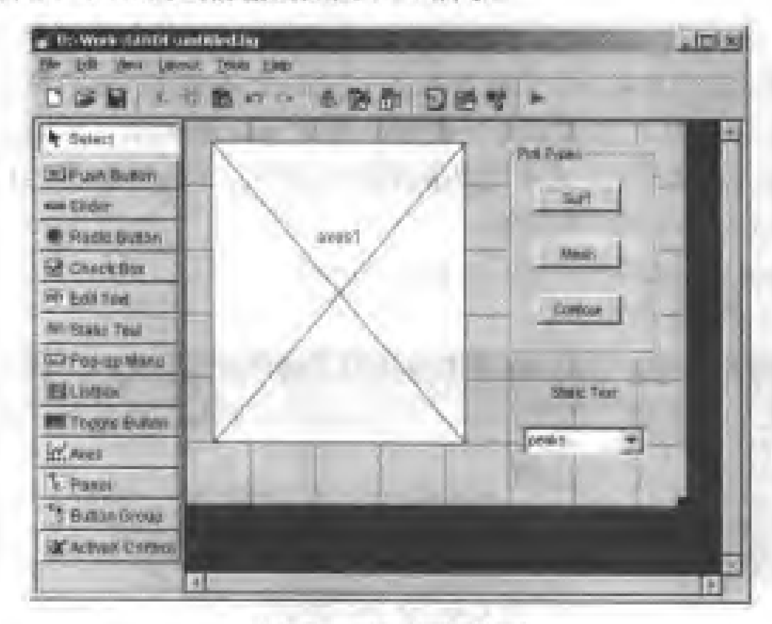


图 5-22 GUI 的当前效果

#### 5．回调属性

组件通过回调进行工作。回调是当用户完成指定动作时执行的函数，这些动作包括单击按钮、选择下拉式列表框中的选项或按下键盘上的键，以及创建和删除组件等。每个组件和下拉式列表框选项都有一个指定回调的属性。创建 GUI 时，必须编写控制 GUI 操作所必

需的回调函数。

一个组件可以有多个回调属性，但是几乎每个组件都有的属性是 Callback 属性。提供给 Callback 属性的代码可以完成组件的主要工作。

6. Tag 属性

Tag 属性将一个字符串作为每个组件的惟一标识符进行提供。GUIDE 使用该标识符为 GUI 中的不同组件创建惟一的回调名称。

第一次在输出编辑器中添加组件时，GUIDE 将 Tag 属性的值设置为默认字符串，例如 pushbutton1。如果组件有一个 Callback 属性，GUIDE 还会将 Callback 属性的值设置为字符串%automatic，如图 5-23 所示。

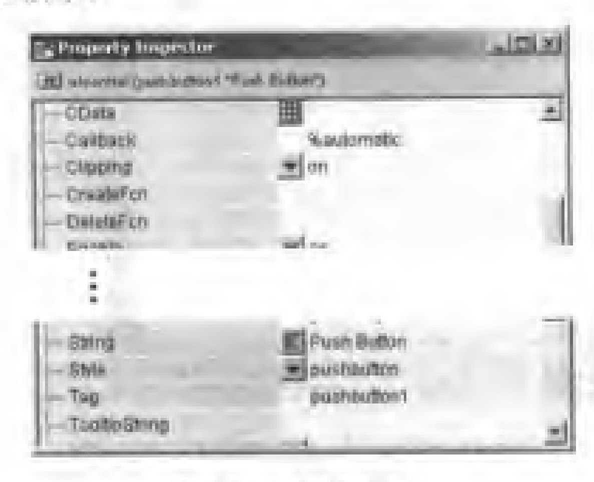



图 5-23　设置 Tag 属性

保存或运行 GUI 时，GUIDE 生成一个 M 文件，该文件为每个组件创建回调函数的框架。GUIDE 给 M 文件中的每个回调函数创建一个惟一的函数名，该名称将组件的 Tag 属性值作为前缀放在字符串“_Callback”前面，例如 pushbutton1_Callback。GUIDE 还将 Callback 属性的值变为字符串，它是回调函数的调用序列。例如，如果 GUI M 文件的名称是 simple_gui，则 Callback 属性的新值为

```
untitled('pushbutton1_Callback',gcbo,[],guidata(gcbo))
```

设置回调属性如图 5-24 中所示。

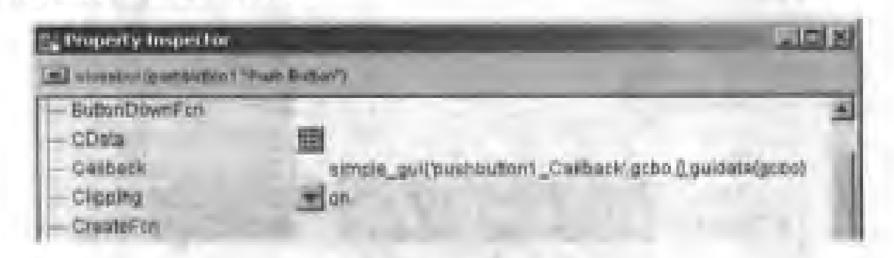



图 5-24　设置回调属性

可以将 Tag 属性的值定义得更具描述性，但是每个 Tag 属性的值对于给定的 GUI 而言必须是惟一的。在下面的例子中，在第一次保存和运行 GUI 前将下拉式列表框的 Tag 属性值改变为 plot_pop-up。图 5-25 中显示了新的 Tag 值。

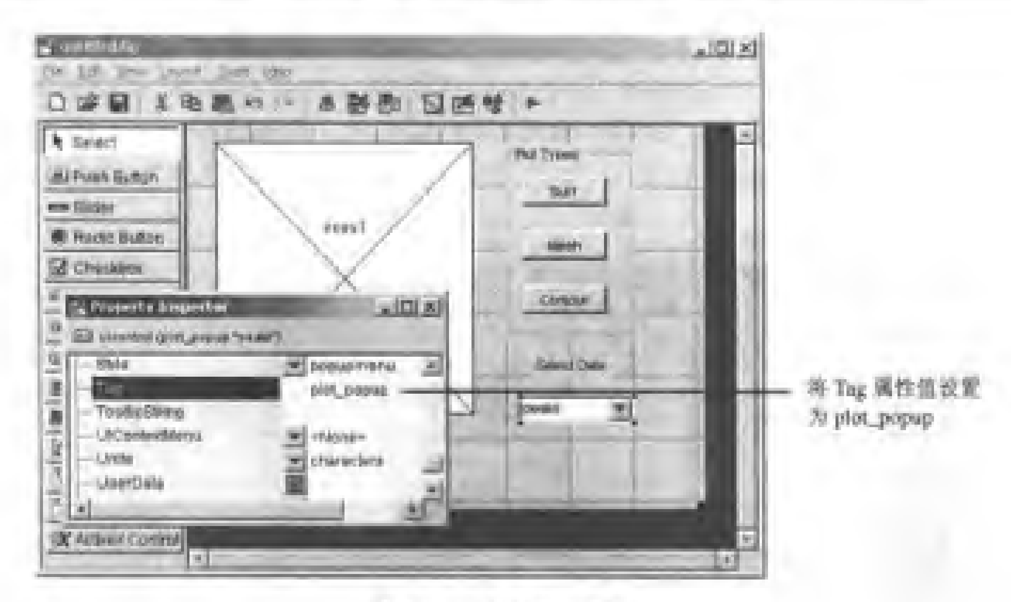



图 5-25　改变 Tag 属性

保存或运行 GUI 时，GUIDE 将下拉式列表框 Callback 属性中回调子函数的名称设置为 plot_pop-up_Callback。如果以后改变 Tag 属性，GUIDE 会更新 Callback 属性，以与新的 Tag 属性值相匹配。

### 5.2.4　GUI 编程

在输出编辑器中完成 GUI 的设计和属性设置以后，下一步工作就是进行编程。其主要内容包括：

- 创建 GUI M 文件；
- 打开 GUI M 文件；
- 在回调间共享数据；
- 在初始化函数中添加代码；
- 在回调中添加代码；
- 用对象浏览器识别回调。

**1．创建 GUI M 文件**

第一次保存或运行 GUI 时，GUIDE 会生成一个函数式 M 文件，它包含每个组件通常要用到的回调函数的框架，以及一些初始化代码，一个初始函数回调和一个输出函数回调。要使回调函数工作，必须在初始框架中添加代码。

可以从“File”菜单中选择“Save”或“Save as”选项实现 GUI 的保存，或者在工具条上单击按钮。可以通过从“Tools”菜单中选择“Run”选项或者在工具条上单击按钮来运行 GUI。

GUIDE 生成 M 文件以后，它打开“Save GUI as”对话框，在“File name”文本框中键入文件名称。GUIDE 将相同的名称指定给 FIG 文件和 M 文件。单击“Save”按钮时，GUIDE 会保存 M 文件并在 M 文件编辑器中打开它。如果是生成本例的 GUI，使用文件名 simple_gui。

### 2．打开 GUI M 文件

下面添加 3 个按钮和下拉式列表框的回调函数代码。一旦 GUIDE 创建了 M 文件，就可以通过在工具条上单击按钮来打开 MATLAB 编辑器。在编辑器中，可以通过单击工具条上的 *f* 按钮，然后在下拉式列表框中选择目标回调来将光标移动到指定的回调函数，如图 5-26 中所示。

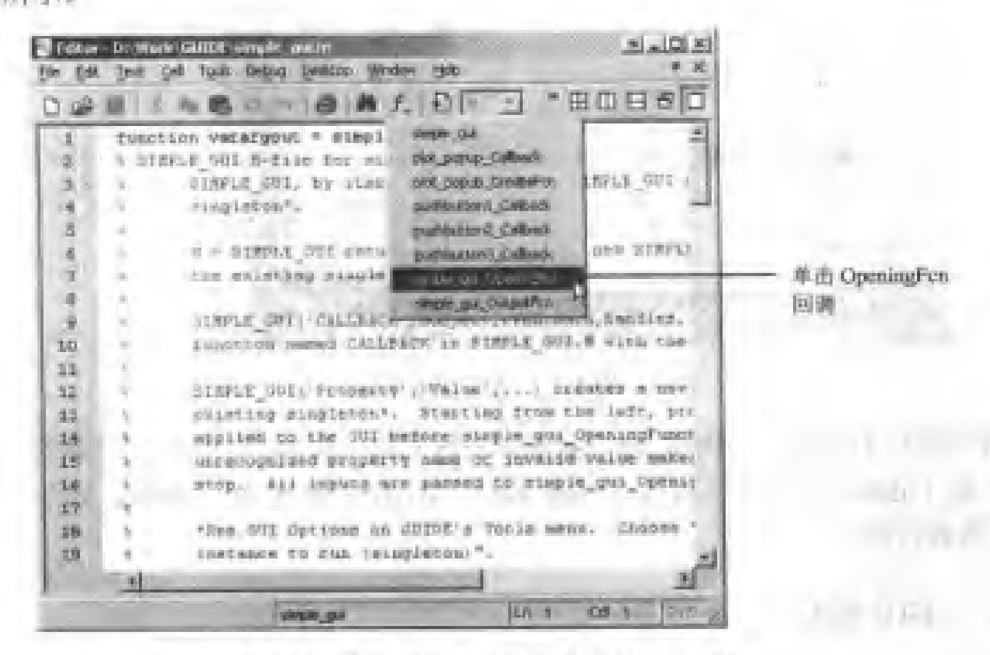



图 5-26　在回调列表中单击 OpeningFcn 回调

例如，单击“simple_gui_openingFcn”将光标移动到初始化函数，如图 5-27 所示。

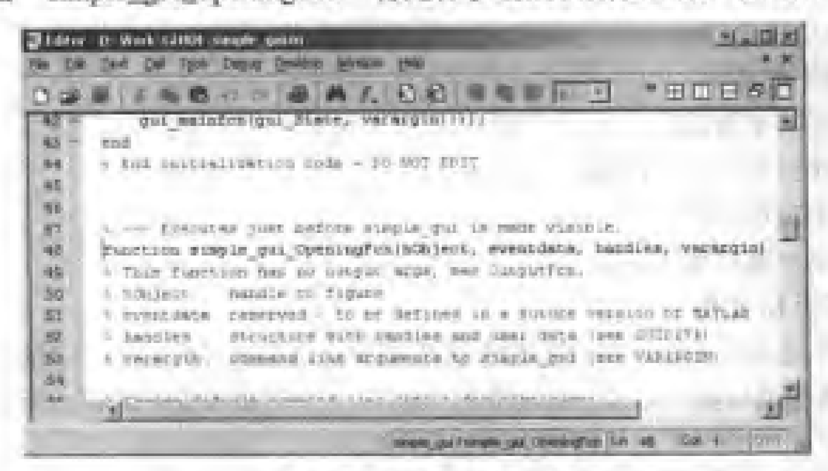


图 5-27　显示初始化函数

### 3．共享回调间的数据

通过将数据保存到 MATLAB handles 结构中，可以实现回调间的数据共享。GUI 中的所有组件使用同一个 handles 结构。它会传递一个输入变量给 GUIDE 生成的所有回调函数。

例如，要将向量 $X$ 中的数据保存到 handles 结构中，按照下面的步骤进行操作：

（1）为 handles 结构的字段选择一个名称，用于保存数据，例如 handles.my_data；

（2）给 handles 结构添加字段并使之等于 $X$，即

```
handles.my_data = X;
```

（3）用 guidata 函数保存 handles 结构，即

```
guidata(hObject,handles)
```

这里，hObject 是执行回调的组件对象的句柄。组件对象的句柄作为输入变量 hObject 传递给 GUIDE 生成的每个回调函数。

要在另一个回调中提取 ***X***，使用下面的命令：

```
X = handles.my_data;
```

可以在任何回调函数中获取 handles 结构中的数据，因为 hObject 和 handles 是 GUIDE 生成的所有回调函数的输入变量。

**4．在初始化函数中添加代码**

在所有 M 文件中，初始化函数是首先调用的。一般用它进行数据初始化和组件默认属性设置。

本例中，在初始化函数中添加代码，用 MATLAB 函数 peaks,membrane 和 sinc 创建 3 套数据集。

注意，GUIDE 用 M 文件的名称加上字符串“_OpeningFcn”来命名初始化函数。例如，本例中 M 文件的名称为 simple_gui.m，所以初始化函数的名称为 simple_gui_OpeningFcn。

下面在初始化函数中添加创建绘图数据的代码。

```
% --- Executes just before simple_gui is made visible.
function simple_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to untitled (see VARARGIN)

% 创建绘图数据
handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
handles.sinc = sinc;
handles.current_data = handles.peaks;
surf(handles.current_data)
```

前 6 个可执行行用 MATLAB 函数 peaks,membrane 和 sinc 生成数据，接下来的一行将 handles 结构的 current_data 字段值设置为 peaks 数据。最后一行显示 peaks 数据的曲面图，当 GUI 第一次打开时，显示该图。GUIDE 会在上面添加的代码后面自动添加两行代码，即

```
handles.output=hObject
guidata(hObject,handles)
```

第一行将句柄保存到 GUI，以便后面在输出函数中可以使用。该命令行在试图将 GUI 句柄返回到命令行是有用。第二行保存 handles 结构。

图 5-28 显示了本例 GUI 第一次运行时的效果。

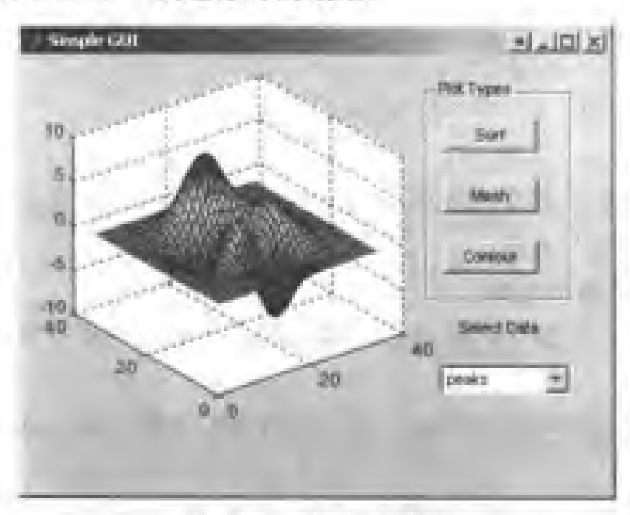



图 5-28　GUI 运行效果

### 5. 添加代码到回调函数

GUI 运行时，如果用户单击界面上的某个组件，例如按钮，MATLAB 会执行组件 Callback 属性指定的回调。本例中，“Surf” 按钮回调函数的名称为 surf_pushbutton_Callback。

（1）按钮的回调函数

“Surf” 按钮的回调函数为 surf_pushbutton_Callback，其代码如下所示。

```
% --- Executes on button press in surf_pushbutton.
function surf_pushbutton_Callback(hObject, eventdata, handles)
% hObject      handle to surf_pushbutton (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Display surf plot of the currently selected data
surf(handles.current_data);
```

类似地，可以给 “Mesh” 按钮和 “Contour” 按钮添加回调函数的代码。

“Mesh” 按钮回调函数的主要代码如下所示。

```
% Display mesh plot of the currently selected data
mesh(handles.current_data);
```

“Contour” 按钮回调函数的主要代码如下所示。

```
% Display contour plot of the currently selected data
contour(handles.current_data);
```

（2）下拉式列表框的回调函数

使用下拉式列表框，允许用户选择绘图数据。每次用户选择这 3 套数据中的一套时，下拉式列表框的回调函数都会读取下拉式列表框的 Value 属性值，以便确定当前绘图数据，并相应地设置 handles.current_data。在 plot_popup_Callback 中函数定义代码后面添加的代码：

```
% --- Executes on selection change in data_popup.
function plot_popup_Callback(hObject, eventdata, handles)
% hObject      handle to surf_pushbutton (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

val = get(hObject,'Value');
str = get(hObject, 'String');
switch str{val};
case 'peaks' % User selects peaks
    handles.current_data = handles.peaks;
case 'membrane' % User selects membrane
    handles.current_data = handles.membrane;
case 'sinc' % User selects sinc
    handles.current_data = handles.sinc;
end
guidata(hObject,handles)
```

（3）用对象浏览器识别回调

本例中，因为程序比较简单，跟踪与每个 GUI 组件对应的回调比较容易。但是，在更加复杂的 GUI 中，跟踪回调会更加困难。为了识别与回调相对应的组件，在输出编辑器中从"View"菜单中选择"Object Browser"选项，或者在工具条上单击按钮，打开图 5-29 所示的对象浏览器。对象浏览器列出了 GUI 每个组件的 tag 和 string 属性。在列表中选择组件名称时在输出编辑器中也会选中该组件。例如，在图 5-29 中，uicontrol 组件（mesh_pushbutton "Mesh"）在对象浏览器中被选中，对应的组件在输出编辑器中也被选中。

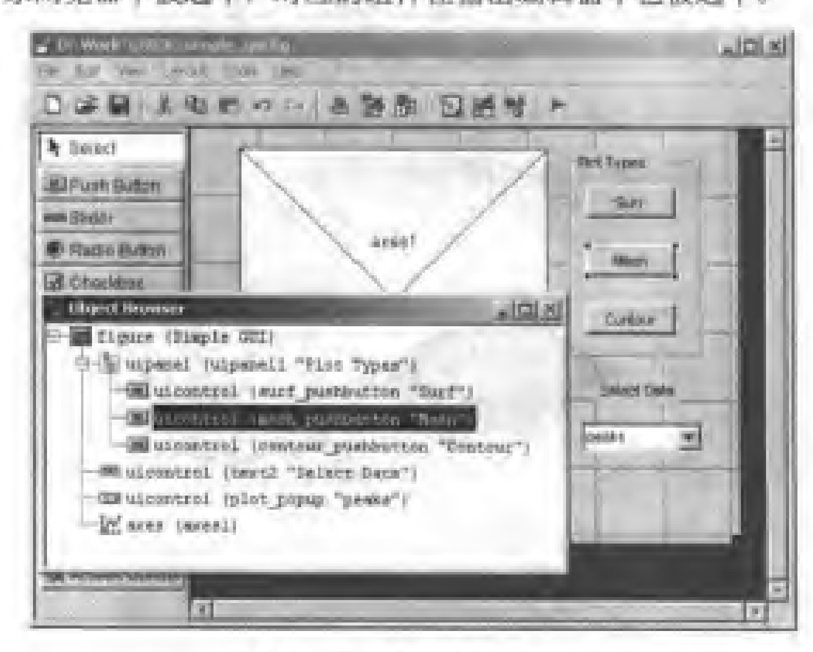



图 5-29　用对象浏览器识别回调

### 5.2.5　保存和运行 GUI

编写回调以后，可以从“Tools”菜单中选择“Run”选项，或者单击 GUIDE 工具条中的▶按钮，运行 GUI。如果最近没有保存 GUI，GUIDE 会显示图 5-30 所示的对话框提醒你。

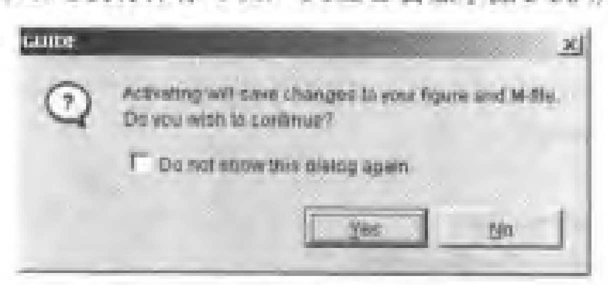



图 5-30　提醒信息

显示此对话框时，单击“Yes”按钮，然后将 GUI 文件保存到一个可写的目录。

如果保存 GUI 的目录不在 MATLAB 的安装路径上，GUIDE 会打开图 5-31 所示的对话框，让你选择是改变当前路径还是将该目录添加到 MATLAB 安装路径上。

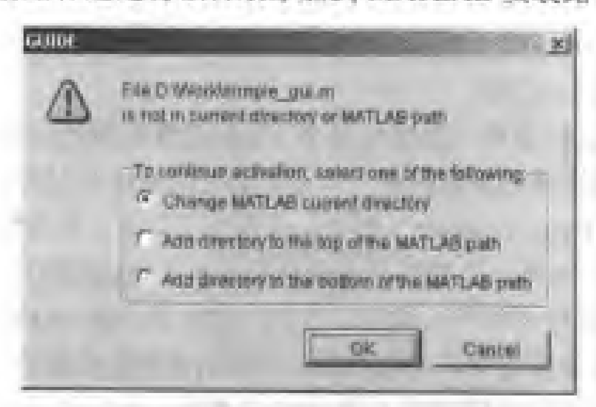



图 5-31　路径选择

单击“OK”按钮，改变当前工作路径。然后 GUIDE 会打开 GUI，如图 5-32 所示。

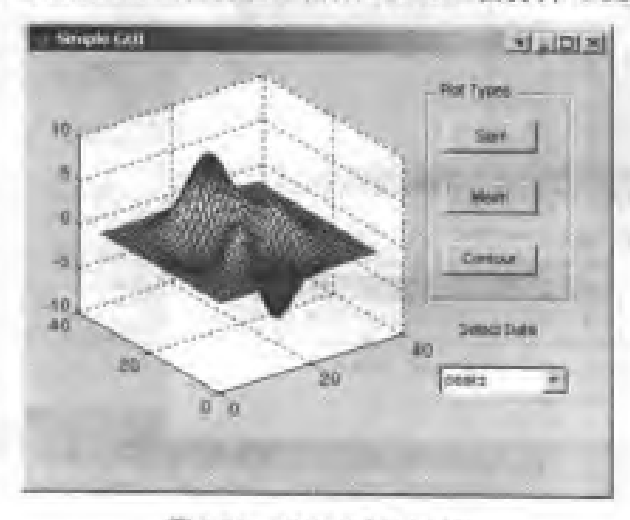



图 5-32　GUIDE 打开 GUI

# 第 6 章　编译和接口

第 4 章介绍了 M 文件。M 文件有很多优点，但是它不能脱离 MATLAB 独立运行。也就是说，在其他机器上运行 M 文件时，必须在该机器上安装对应版本的 MATLAB 软件。在有些情况下，这是不方便的。使用 MATLAB 编译器，可以将大部分 M 文件编译为独立应用程序。MATLAB 7.0 采用最新的 4.0 版编译器。

另外，MATLAB 与其他应用程序接口，可以起到互通有无，取长补短的效果。本章主要介绍 MATLAB 7.0 与 Visual Basic 的接口，介绍最新的组件生成工具—COM 生成器 1.1。

## 6.1　MATLAB 编译器 4.0

MATLAB 7.0 采用的这款编译器版本为 4.0，相对于以前诸版本有所改进。本节介绍编译器 4.0 的特点和使用方法。

### 6.1.1　MATLAB 编译器 4.0 的新特点

编译器 4.0 用于创建可分发给其他用户的应用程序和库。这个版本的编译器支持 MATLAB 的所有特性，包括对象数据类型。下面具体进行介绍。

- 编译器 4.0 使用新的 MATLAB 组件运行时（MCR），而不是 MATLAB C/C++数学和图形库。MCR 是一个独立的共享库，利用它可以运行 MATLAB 编译器创建的经过加密的 M 文件；
- 编译器 4.0 只为接口函数生成代码，而以前各版本的编译器会给整个 M 文件生成代码；
- 编译器 4.0 具有与代码生成和格式化有关的取消选项；
- 编译器 4.0 取消了某些打包选项及与它们相关的打包文件；
- 编译器 4.0 支持的操作系统只有 Microsoft Windows 和 Linux；
- 编译器 4.0 包括了几个新选项；
- 编译器 4.0 不包括针对 Visual Studio 的 MATLAB 插件；
- 编译器 4.0 不会加速程序的运行。运行编译后的应用程序与在 MATLAB 中运行没有速度上的差别；
- 编译器 4.0 不支持 MEX 文件和 Simulink S 函数；
- MATLAB 不支持用 loadlibrary 函数载入 MATLAB 编译器生成的库。

### 6.1.2　MATLAB 编译器的使用

MATLAB 编译器可以生成独立应用程序、库、COM 对象和 Excel 插件。

1. 包装器文件

MATLAB 编译器（mcc）根据 M 文件生成可以分发的独立应用程序或软件组件。最终生成的结果可以是独立应用的库或组件中的任何一种类型。编译器根据最后要生成的目标类型生成合适的包装器文件。包装器文件包含编译后应用程序和所支持的可执行对象类型之间的接口。

包装器文件根据执行环境的不同而有所不同。为了提供必要的接口，包装器具有下面一些功能：

- 完成包装器指定的初始化和终止运行工作；
- 定义包含路径信息、加密密匙和 MATLAB 组件运行时(MCR)所需要的其他信息的数组；
- 提供传递接口函数调用 MCR 中 MATLAB 函数时的必要代码。

例如，包装器是一个包含 main 函数的独立应用。库的包装器包含每个公共 M 文件函数的入口点。

MATLAB 编译器生成的组件技术文件（CTF）与最后生成的目标类型（可执行程序或库）是独立的。包装器文件会提供与目标类型之间的必要接口。

2. 独立应用

用-m 选项调用时，MATLAB 编译器使用输入的 M 文件，生成适合于独立应用的必要的包装器文件。然后，C 或 C++ 编译器编译该代码并连接 MCR。例如，下面的代码生成文件 example.m 的独立可执行文件

```
mcc -m example
```

3. 库

可以用-l 选项根据 M 文件集生成 C 共享库。例如，

```
mcc -l file1.m file2.m file3.m
```

用 file1.m,file2.m 和 file3.m 生成一个共享库。-l 选项是一个打包选项，可以扩展为

```
-w lib -T link:lib
```

-w lib 选项让 MATLAB 编译器为共享库生成一个函数包装器，并调用它 libfile1。-T link:lib 选项把目标输出指定为共享库。

4. COM 生成器和 Excel 生成器

使用可选的 MATLAB COM 生成器，可以创建 COM 组件。该组件可以用于任何操作 COM 对象的应用程序。MATLAB COM 生成器用编译器根据 M 文件创建组件对象模型（COM），M 文件集合被转换为 COM 类。COM 生成器支持一个组件中有多个类的情况。

利用可选的 Excel 生成器，可以自动生成 Visual Basic 应用文件（.bas）和插件 DLL。可以在 Excel 中直接使用 .bas 文件和导入插件。MATLAB Excel 生成器将 MATLAB M 文件编译为可以用作 Excel 插件的 COM 对象，M 文件集合被转换为单个的 Excel 插件。MATLAB Excel 生成器支持一个组件中有一个类。

### 6.1.3 编译独立应用程序

根据 M 文件 mymfunction 创建独立应用程序，使用下面的命令：

```
mcc -m mymfunction.m
```

这将创建一个名为 myfunction.exe 的独立可执行程序。

根据 M 文件 mymfunction 创建共享库，使用下面的命令：

```
mcc -l mymfunction.m
```

这将创建一个名为 mymfunction.dll 的共享库。

下面提供一个实例演示如何编译应用程序并将它分发到其他机器上。用 M 文件 magicsquare.m 创建独立 C 应用 magicsquare。

（1）编译应用程序

- 将 magicsquare.m 从下面的目录中复制到工作目录（MATLAB 安装目录下的 work 目录）上，其中<matlabroot>表示 MATLAB 的安装目录，后面类同。

```
<matlabroot>/extern/examples/compiler
```

- 用下面的语句编译 M 代码。

```
mcc -mv magicsquare.m
```

其中，-m 选项让 MATLAB 编译器（mcc）生成一个 C 独立应用。-v 选项显示编译过程中的各个步骤，并帮助识别其他有用信息，例如使用了哪些第三方编译器，引用了哪些环境变量等。

该命令创建一个名为 magicsquare 的独立应用和其他文件。

（2）测试应用程序

按照下面的步骤测试刚刚创建的独立应用：

- 将下面的目录添加到动态库路径中。

```
<matlabroot>\bin\win32
```

- 运行独立应用程序 magicsquare.exe。

```
magicsquare.exe 4
ans =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

（3）分发应用

可以将 MATLAB 编译器生成的独立应用分发到任何目标机器上，只要该机器使用的操作系统与编译独立应用的机器所使用的操作系统相同。下面介绍分发应用的步骤：

- 在开发机器上生成 MATLAB 组件运行时（MCR）库文档。只需要在每个平台上进行一次就可以了。生成 MCR 库文档，运行下面的语句。

```
buildmcr;
```

该语句将 MCRInstaller 文档 MCRInstaller.zip 放在<matlabroot>/toolbox/compiler/deploy/<arch>目录下。

也可以指定文件名和路径，使用下面的语句。

```
buildmcr(path,filename);
```

其中，path 为文档保存路径，filename 为文件名。

要把完整路径返回到文件 zipfile，使用下面的语句。

```
zipfile = buildmcr(path, filename);
```

要在当前目录中生成 MCRInstaller 文档，使用下面的语句。

```
zipfile = buildmcr('.');
```

● 搜集和打包表 6-1 所示的文件，并分发到目标机器上。

表 6-1　打包文件

| 打包文件 | 描述 |
|---|---|
| MCRInstaller.zip | (Linux) MATLAB 组件运行时库文档；用户所用的平台必须与开发平台一致 |
| MCRInstaller.exe | (Windows) 自解压 MATLAB 组件运行时库工具；用户所用的平台必须与开发平台一致 |
| unzip | (Linux) 解压 MCRInstaller.zip 文件的工具，目标机器上必须安装了解压工具 |
| magicsquare.ctf | 组件技术文件；用户所用的平台必须与开发平台一致 |
| magicsquare | 应用程序 magicsquare.exe |

（4）运行应用程序

终端用户必须按照下面的步骤安装和运行应用程序：

● 通过在目录中运行 MCR 安装器来安装 MCR。例如，在 C:\MCR 中运行 MCRInstaller.exe。将组件和 CTF 文档复制到应用的根目录下，例如，C:\approot。然后将下面的目录添加到动态库路径中，其中<mcr_root>表示 MCR 的安装路径：

```
<mcr_root>\runtime\win32
```

● 运行应用程序。从系统提示符运行 magicsquare 独立应用并提供魔方矩阵的大小参数。

```
magicsquare 4
```

显示下面的结果：

```
ans =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

创建独立应用的途径之一是把所有源代码写成一个或多个 M 文件或 MEX 文件，就像前面的魔方示例中的那样。将应用按 M 文件编码可以利用 MATLAB 交互开发环境的好处。一旦程序的 M 文件版本可以正常工作，就可以编译代码并把它生成为独立应用。

考虑一个简单的应用，它的源代码由两个 M 文件组成，即 mrank.m 和 main.m。下面根据它们创建 C 代码。

（1）mrank.m

mrank.m 返回一个整型向量 ***r***，***r*** 的每一个元素都表示魔方的秩。例如，函数运行以后，r(3)包含 3×3 魔方矩阵的秩。

```
function r = mrank(n)
r = zeros(n,1);
for k = 1:n
    r(k) = rank(magic(k));
end
```

本例中，语句行 r=zeros(n,1)通过预分配内存来帮助改善编译器的运行。

（2）main.m

main.m 包含一个主过程，它调用 mrank 函数，然后显示结果。

```
function main
r = mrank(5)
```

（3）编译示例

调用 MATLAB 编译器，将上面的 M 文件编译成独立应用可用的代码。

```
mcc -m main mrank
```

-m 选项会导致 MATLAB 编译器生成适合于独立应用的 C 源代码。例如，MATLAB 编译器生成 C 源代码文件 main.c,main_main.c 和 mrank.m。main_main.c 包括一个名为 main 的 C 函数，main.c 和 mrank.c 包含名为 mlfMain 和 mlfMrank 的 C 函数。

要生成一个可执行程序，可以用 mbuild 命令编译和链接这些文件。或者，用下面的语句自动完成整个过程（对两个 M 文件都调用 MATLAB 编译器，使用 mbuild 命令，用 ANSI C 编译器编译这些文件，并链接代码）。

```
mcc -m main mrank
```

如果需要组合其他代码，或者想生成编译应用的工具文件，使用下面的命令。

```
mcc -mc main mrank
```

-c 选项限制 mbuild 的调用。

## 6.2　MATLAB 与 Visual Basic 接口

MATLAB 与 Visual Basic 接口可以有多种手段，从传统的 DDE,OLE 到 ActiveX 控件、COM 组件技术等都可以使用。文献 6 中对于 MATLAB 与 Visual Basic 接口有比较详细的介绍，可以参阅。这里主要介绍 MATLAB 最新的 COM 生成器 1.1，利用它生成 COM 组件，并实现与 Visual Basic 的接口。COM 组件技术是目前比较流行的技术，MATLAB 从 6.5 版本开始提供 COM 生成器，版本为 1.0。

### 6.2.1　COM 生成器 1.1

用 MATLAB COM 生成器创建 COM 组件是一个简单的过程，只需要 4 个步骤，即创建工程、管理 M 文件和 MEX 文件、生成组件和打包分发组件。

**1．创建工程**

在 MATLAB 命令行中输入命令 comtool，打开“MATLAB COM Builder”对话框，如图 6-1 所示。它是 MATLAB COM 生成器的主要工作环境。

在“File”菜单中选择“New Project”选项，打开“New Project Settings”对话框，如图 6-2 所示。

在“Component name”文本框中输入组件（DLL 文件）的名称。输入组件名以后，生成器自动在“Class name”文本框中输入与组件名相同的名称。用户可以根据需要进行修改。尽管组件名和类名可以相同，但组件名不能与任何 M 文件或 MEX 文件的文件名相同。

要把类添加到组件中，在“Class name”文本框中输入类名，并单击“Add>>”按钮。添加的类显示在“Classes”列表中。

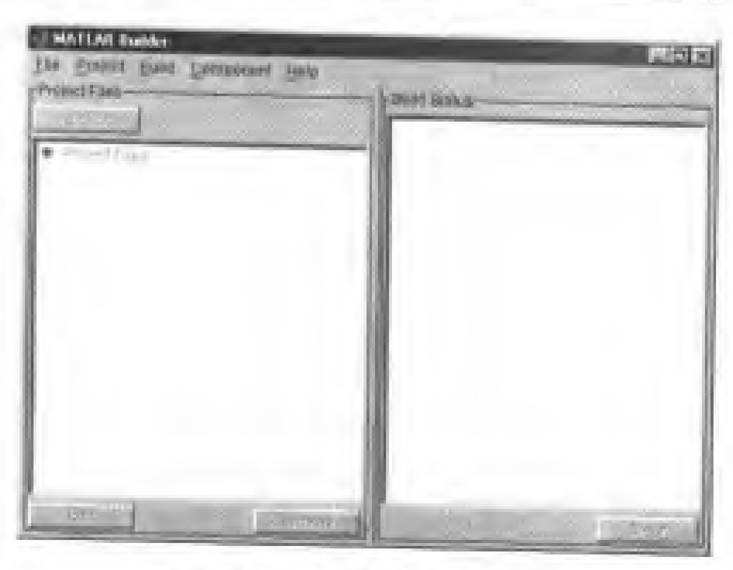


图 6-1　MATLAB COM 生成器主窗口

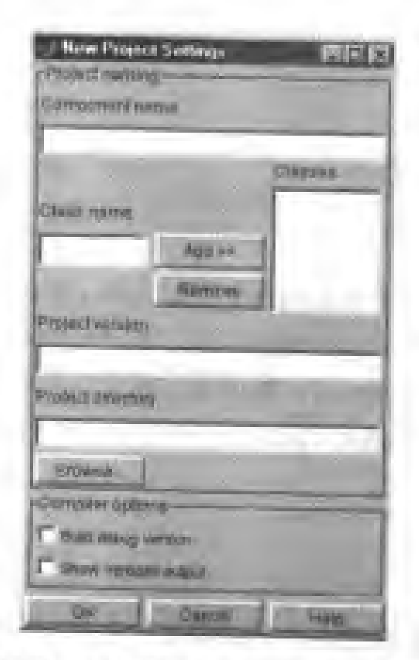


图 6-2　"New Project Settings" 对话框

"Project version" 文本框中设置组件的版本号，默认时组件版本号为 1.0。

"Project directory" 文本框指定编译和打包模型时工程和相关文件的存放位置。工程目录由当前目录和组件名自动组合生成。

可以创建编译模型的一个调试版本，并能在调用 MATLAB 编译器时指定详细输出。该调试版本一方面允许跟踪，使错误报告显示 M 文件和发生错误的行。所有跟踪信息都会进行报告。如果不进行调试，就得不到 MATLAB 代码中发生错误的位置的指示。另一方面，它允许使用 Visual Studio 调试器进行完整的调试。

设置完以后，单击 "OK" 按钮，它们就成为工程工作空间的一部分并与添加到工程中的所有 M 文件和 MEX 文件一起被保存到工作空间中。一个名为<component_name>.cbl 的工程文件被自动添加到工程目录中。

### 2. 管理 M 文件和 MEX 文件

创建工程以后，生成器主窗口中的 "Project"，"Build" 和 "Component" 等 3 个菜单就变为可用。如图 6-3 所示。

单击 "Add File" 按钮或从 "Project" 菜单中选择 "Add File..." 选项，向工程中添加 M 文件和/或 MEX 文件。一次只能添加一个文件。

单击 "Remove" 按钮或从 "Project" 菜单中选择 "Remove" 选项，删除所有选定的 M 文件或 MEX 文件。可以选择多个文件，然后一次删除。




图 6-3　选项被激活的主窗口

选择 M 文件以后单击“Edit”按钮，或从“Project”菜单中选择“Edit File...”选项，或直接双击 M 文件名，都可以在 MATLAB 编辑器中打开该 M 文件并进行编辑和调试。不能编辑 MEX 文件。

**3．生成组件**

定义工程设置并添加必要的 M 函数和 MEX 函数以后，可以通过选择“Build”菜单中的“COM Object”选项来调用 MATLAB 编译器，把中间源文件写到<project_dir>\src 目录中，将必要的输出文件写到<project_dir>\distrib 目录中。

“Build Status”面板显示生成过程中的输出，通知遇到的任何问题。出现在<project_dir>\distrib 目录中的将是一个 DLL 文件。生成的 DLL 文件自动注册到系统中。

选择“Build”菜单中“Clear Status”选项，清除“Build Status”面板。生成过程的输出保存在文件<project_dir>\build.log 中。要打开该日志文件，在“Build”菜单中选择“Open Build Log”选项。该文件提供了一个生成过程的记录，在清除“Build Status”面板以后，可以引用该记录。

**4．打包和分发组件**

一旦模型编译成功并且进行了测试，就可以把它打包并分发给终端用户了。从“Component”菜单中选择“Package Component”选项，创建一个包含表 6-2 所示的文件的自解压可执行程序。

**表 6-2　自解压可执行程序包含的文件**

| 文　件 | 功　能 |
|---|---|
| _install.bat | 由自解压可执行程序运行的脚本 |
| <componentname_projectversion>.dll | 编译后的组件 |
| mglinstaller.exe | MATLAB 数学库和图形库安装器 |
| mwcomutil.dll | COM 生成器工具库 |
| mwregsvr.exe | 在计算机上注册 DLL 的可执行程序 |

该自解压可执行程序的名称为<componentname>.exe。在计算机上运行安装器，进行以

下步骤：

- mglinstaller 安装 MATLAB C/C++ 数学和图形库；
- 添加 mglinstaller 创建的目录<application>\bin\win32（<application>表示配置应用的根目录）；
- mwregsvr 注册 mwcomutil.dll 和<componentname>_<projectversion>.dll。

### 6.2.2　用 COM 生成器生成组件

下面结合一个例子来介绍 COM 生成器的使用。本例用 MATLAB 提供的 fitfun 函数演示如何用非线性函数拟合数据集。fitfun 函数用 fminsearch 函数最小化多变量的非线性函数。fminsearch 函数实现了 Nelder-Mead 单纯形算法。

这个例子演示了如何用 COM 生成器创建 COM 组件和如何在 Visual Basic 中使用这个 COM 组件。

**1．创建工程**

将当前目录设置为 MATLAB 安装目录下的 work 目录。在命令窗口键入 comtool 命令，启动 COM 生成器图形用户界面。

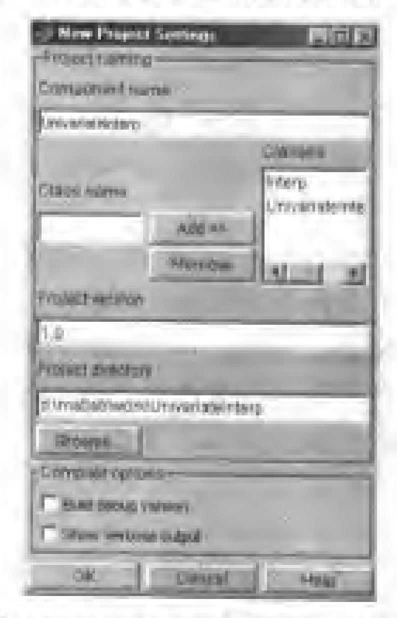


图 6-4　“New Project Settings”对话框

在“File”菜单中选择“New Project”选项，打开“New Project Settings”对话框。在该对话框中，进行如下设置：

- 在“Component name”文本框中输入组件名 CurveFit。单击“Tab”键将鼠标光标移动到“Class name”文本框；
- COM 生成器会自动在“Class name”列表框中输入类名 CurveFit；
- 默认时版本号为 1.0，不改变它；
- “Project directory”文本框中包含 COM 生成器启动的默认目录<matlabroot>\work，以及组件名 CurveFit。可以将它改变为任何目录。如果所选择的目录不存在，系统会提醒你创建它。现在“New Project Settings”对话框如图 6-4 所示；
- 单击“OK”按钮，创建 CurveFit 工程。

**2．生成组件**

按照下面的操作步骤生成组件：

（1）在 COM 生成器图形用户界面中单击“Add File”按钮；

（2）在目录<matlab>\work\CurveFitDemo 中添加文件 fitfun.m 和 fitdemo.m，如图 6-5 所示。

（3）在“Build”菜单中选择“COM Object”选项，创建组件。完成创建以后的 COM 生成器图形用户界面如图 6-6 所示。

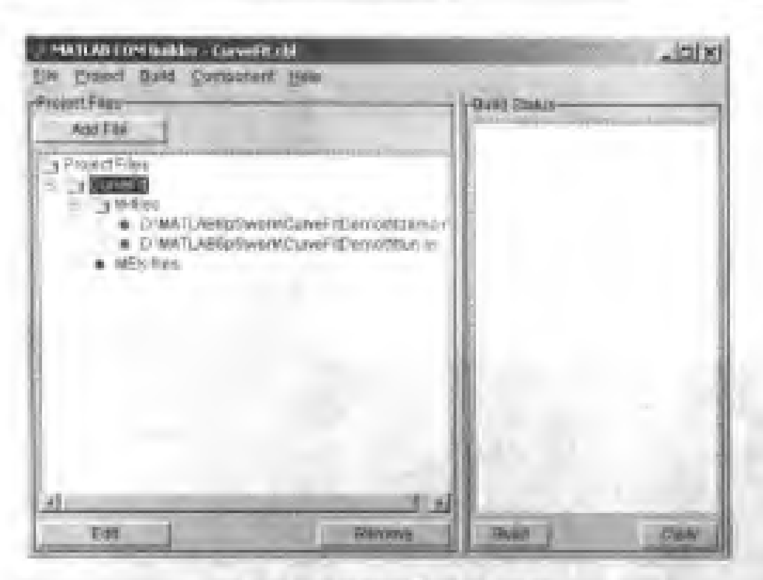


图 6-5　在工程中添加 M 文件



图 6-6　完成组件创建以后的 COM 生成器图形用户界面

创建的组件放在类目录的 distrib 目录下。

### 6.2.3　在 Visual Basic 中使用组件

可以在任何支持 COM 机制的应用程序中调用前面生成的组件。按照下面的步骤创建一个 Viaual Basic 工程并添加库引用。

（1）启动 Visual Basic；

（2）创建一个新的“标准 Exe”工程，如图 6-7 所示的是 Visual Basic 编程环境。

（3）在“工程”菜单中单击“引用...”选项，并选择下面的类库：

CurveFit 1.0 Type Library

MWComUtil 1.0 Type Library

下一步创建应用程序的窗体界面，从该窗体获取用户输入的数据。按照以下的步骤创建一个新的用户窗体并添加必要的控件。

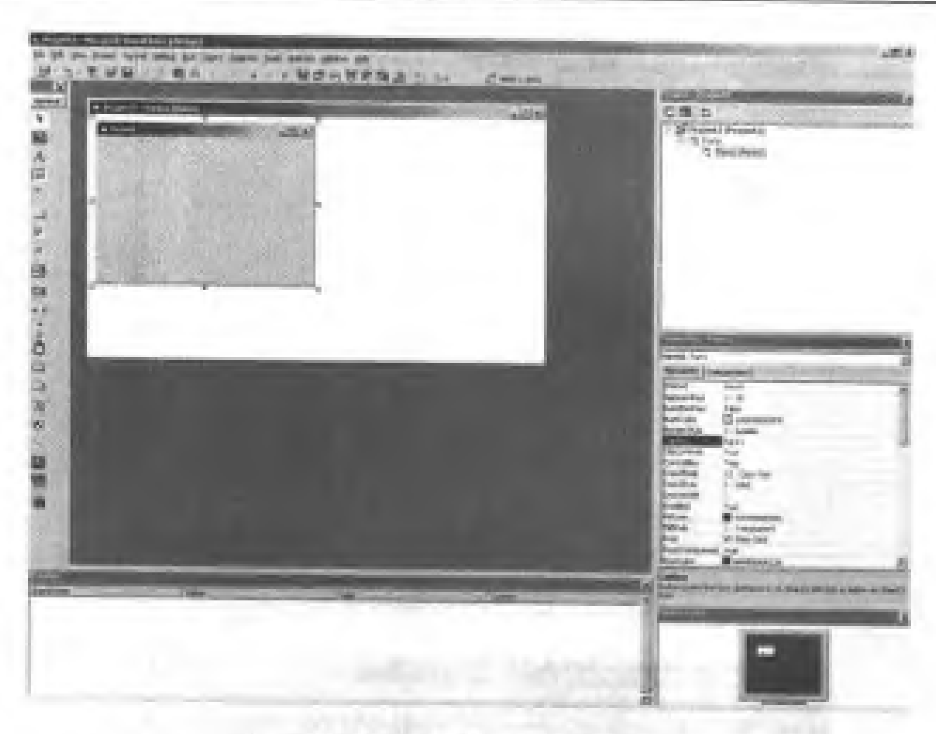


图 6-7　Visual Basic 编程环境

（1）在“Projects”菜单中单击“Component”选项，并选择“Microsoft Windows Common Controls”选项，将从这个组件库中使用 ListView 控件。

（2）现在在空白窗体上添加一系列控件，如图 6-8 所示。

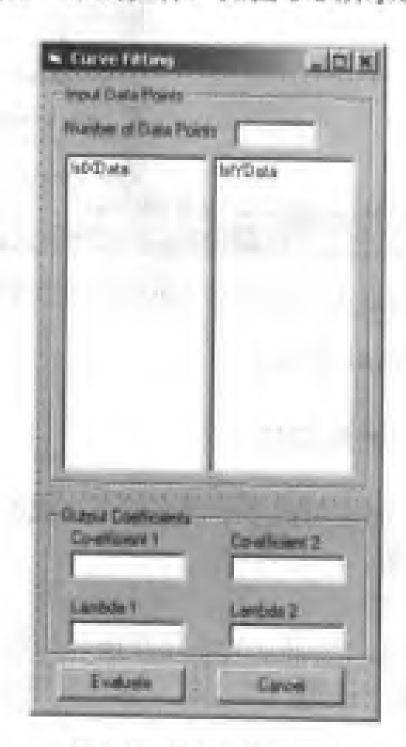



图 6-8　程序的设计界面

各控件的属性设置如表 6-3 中所示。

表 6-3　控件属性设置

| 控件类型 | 控件名称 | 属　性 |
|---|---|---|
| Form | frmCurveFit | Caption = Curve Fitting |
| Frame | frmInput | Name = frmInput*<br>Caption = Input Data Points |
| | frmOutput | Name = frmOutput*<br>Caption = Output Coefficients |
| ListView | lstXData | Name = lstXData<br>GridLines = TrueLabel<br>Edit = lvwAutomatic<br>View = lvwReport |
| | lxtYData | Name = lstYData<br>GridLines = TrueLabel<br>Edit = lvwAutomatic<br>View = lvwReport |
| Label | lblNumDataPoints | Caption = Number of Data Points |
| | lblCoeff1* | Caption = Co-efficient 1 |
| | lblCoeff2 | Caption = Co-efficient 2 |
| | lblLambda1* | Caption = Lambda 1 |
| | lblLambda2 | Caption = Lambda 2 |
| TextBox | txtNumOfDatPoints | Text = |
| | txtCoeff1 | Text = |
| | txtCoeff2 | Text = |
| | txtLambda1 | Text = |
| | txtLambda2 | Text = |
| CommandButton | cmdEvaluate | Caption = Evaluate<br>Default = True |
| | cmdCancel | Caption = Cancel<br>Cancel = True |

（3）设计部分完成以后，在“File”菜单中单击“Save”选项，保存工程。把CurveFitExample.vbp 和 frmCurveFit.frm 作为工程名和窗体名。

（4）在代码窗口键入下面的代码。

```
Dim theFit As CurveFit.CurveFit ' 保存 COM 对象的变量


' 窗体初始化
Private Sub Form_Initialize()
On Error GoTo Exit_Form
    ' 创建 COM 对象
    Set theFit = New CurveFit.CurveFit
    theFit.MWFlags.ArrayFormatFlags.TransposeOutput = True
    Exit Sub
Exit_Form:
    ' 如果对象创建失败，显示错误信息并卸载窗体
```

```
    MsgBox ("Error: " & Err.Description)
    MsgBox ("Error: Could not create the COM object")
    Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo Exit_Form
    ' 设置组件的运行时属性

    ' 设置列的表头
    Call lstXData.ColumnHeaders.Add(, , "X Data")
    Call lstYData.ColumnHeaders.Add(, , "Y Data")

    ' 设置 LabelEdit 属性
    lstXData.LabelEdit = lvwAutomatic
    lstYData.LabelEdit = lvwAutomatic

    ' 使列表框中的网格线可见
    lstXData.GridLines = True
    lstYData.GridLines = True
    Exit Sub
Exit_Form:
    MsgBox ("Error: Could not load the form")
    MsgBox ("Error: " & Err.Description)
    Unload Me
End Sub

Private Sub cmdCancel_Click()
    ' 如果用户单击取消按钮，卸载窗体
    Unload Me
End Sub

Private Sub txtNumOfDataPoints_Change()
    ' 如果用户改变数据点的个数，清除 XData 和 YData
    Dim loopCount As Integer
    Call lstXData.ListItems.Clear
    Call lstYData.ListItems.Clear
    If (txtNumOfDataPoints.Text = "") Then
        Exit Sub
    End If
    For loopCount = 1 To CInt(txtNumOfDataPoints.Text)
        lstXData.ListItems.Add (loopCount)
```

```
            lstYData.ListItems.Add (loopCount)
        Next loopCount
    End Sub

    Private Sub cmdEvaluate_Click()
        Dim loopCount As Integer
        Dim numOfData As Integer
        Dim XData() As Double
        Dim YData() As Double
        Dim Coeff As Variant
        Dim Lambda As Variant

    On Error GoTo Handle_Error
        If txtNumOfDataPoints.Text = "" Then
            Exit Sub
        End If
        ' 获取数据点数
        numOfData = CInt(txtNumOfDataPoints.Text)
        ReDim XData(1 To numOfData) As Double
        ReDim YData(1 To numOfData) As Double
        ' 将输入数据读入各自的 double 型数组中
        For loopCount = 1 To numOfData
            XData(loopCount) = lstXData.ListItems.Item(loopCount)
            YData(loopCount) = lstYData.ListItems.Item(loopCount)
        Next loopCount

        ' 调用 COM 方法
        Call theFit.fitdemo(2, Coeff, Lambda, XData, YData)

        ' 显示 COM 方法返回的系数值
        txtCoeff1.Text = CStr(Format(Coeff(1, 1), "##.####"))
        txtCoeff2.Text = CStr(Format(Coeff(1, 2), "##.####"))
        txtLambda1.Text = CStr(Format(Lambda(1, 1), "##.####"))
        txtLambda2.Text = CStr(Format(Lambda(1, 2), "##.####"))
        Exit Sub
    Handle_Error:
        MsgBox ("Error: " & Err.Description)
    End Sub
```

然后运行程序即可。利用“File”菜单中的选项，还可以将程序打包为独立应用。

### 6.2.4 使用 COM 生成器时可能遇到的问题

表 6-4 中列出了使用 COM 生成器时可能遇到的一些问题，以及引起问题的原因和解决问题的办法。

**表 6-4　使用 COM 生成器引起的问题和解决办法**

| 出错信息 | 引起问题的原因 | 解决问题的办法 |
| --- | --- | --- |
| MBUILD.BAT: Error: The chosen compiler does not support building COM objects. | 所选择的编译器不支持生成 COM 对象 | 重新运行 mbuild-setup，并选择一种编译器 |
| Error in component_name.class_name.1_0: Error getting data conversion flags. | 一般是因为没有注册 mwcomutil.dll | 打开一个 DOS 窗口，将目录改变为 <matlabroot>\bin\win32，并运行命令 mwregsvr mwcomutil.dll |
| Error in VBAProject: ActiveX component can't create object. | 1. 没有注册工程 DLL<br>2. 在系统路径上的某个地方存在不兼容的 MATLAB DLL | 如果 DLL 没有注册，打开一个 DOS 窗口，将路径改变为 <projectdir>\distrib，<projectdir>表示工程文件的位置，并运行下面的命令<br>mwregsvr <projectdll>.dll. |
| Error in VBAProject: Automation error The specified module could not be found. | MATLAB 没有位于系统路径上 | 将 <matlabroot>\bin\win32 放在路径上 |
| LoadLibrary("component_name_1_0.dll") failed - The specified module could not be found. | 从 DOS 提示符中注册工程 DLL 时，MATLAB 不在系统路径上 | 将 <matlabroot>\bin\win32 放在路径上 |
| Cannot recompile the M file xxxx because it is already in the library libmmfile.mlib. | 所选择的 M 文件的名称与库中 M 文件的名称相同 | 重命名 M 文件，选择一个不与库中 M 文件冲突的名称 |

# 第 7 章　二维图形绘制

MATLAB 提供了很多现成的绘图函数。利用它们，可以绘制线形图、条形图等基本图形，也可以绘制不同坐标系下的图形如对数坐标图、极坐标图等，还可以绘制不同专业用到的功能图形，如玫瑰花图、阶梯图等。

## 7.1　线形图、条形图和面积图

这几种图形在功能上比较相似，都是以直观的形式表示数据的分布特征。绘制线形图、条形图、面积图分别使用 plot 函数、bar 函数和 area 函数。下面结合两个例子介绍这几种图形的生成。

```
x=[1 2 3 4 5 6];
y=[10 15 8 20 22 32];
bar(x,y);
colormap cool;
hold on
plot(x,y, 'LineWidth',3,'Color', 'r', 'LineStyle', '--');
hold off
```

生成图 7-1。

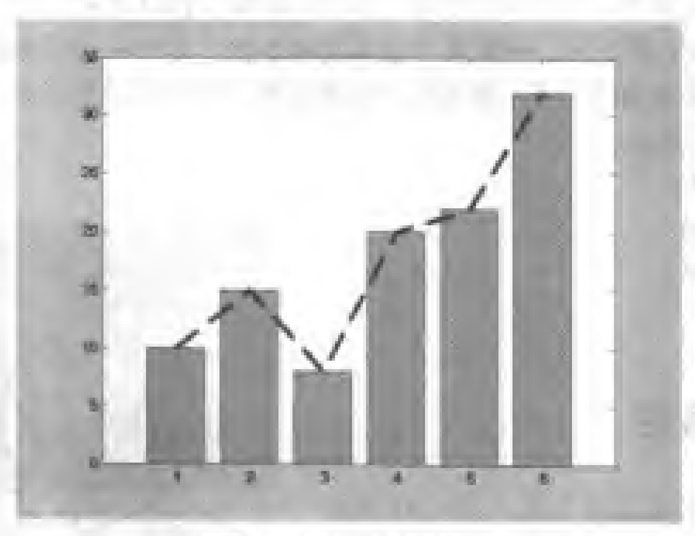



图 7-1　条形图叠加线形图

下面的代码在面积图上叠加条形图。

```
x=[1;2;3;4];
y=[1 5 3;
   3 2 7;
```

```
    1 5 3;
    2 6 1];
area(y);
hold on
bar(x,y);
colormap summer
axis auto
```

生成图 7-2。

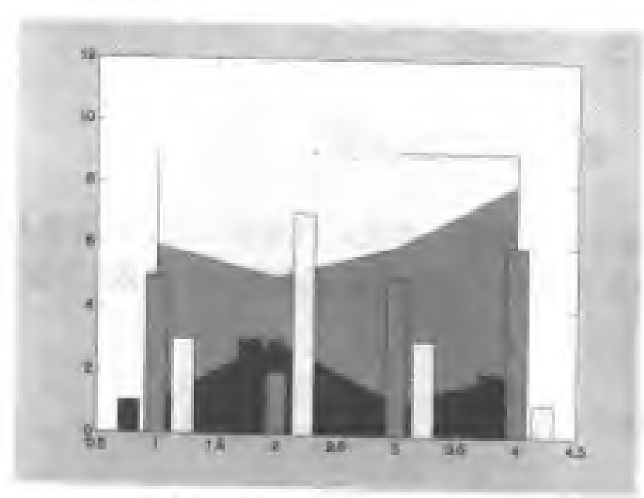


图 7-2　在面积图上叠加条形图

## 7.2　饼图

饼图显示部分与整体之间的比例关系。用 pie 函数和 pie3 函数绘制二维、三维饼图。

下面用 pie 函数生成饼图。向量 *X* 中是某年内 3 种产品的年销量数据，用饼图显示 3 种产品的销量分别占总销量的比例。将最后一年销量对应的区块分离显示。

```
x=[19.3 22.1 51.6];
explode=[0 0 1];
pie(x,explode)
colormap cool
```

生成图 7-3。

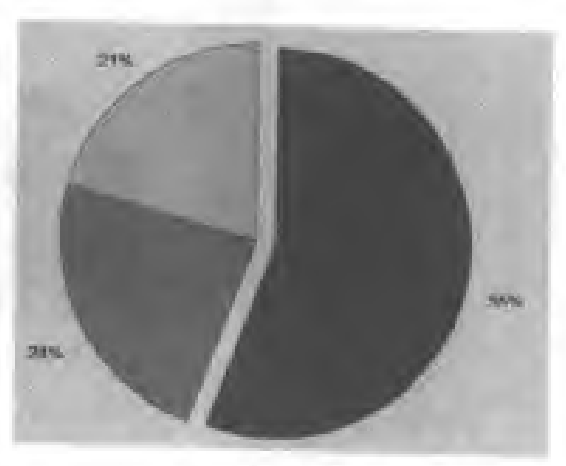



图 7-3　饼图

## 7.3　误差条图

误差条图显示数据的置信区间或沿曲线的偏差。用 errorbar 函数绘误差条图，该函数的调用格式为：

- errorbar(Y,E)　根据 *Y* 的数据绘图并在 *Y* 的每个元素处绘一误差条。误差条两端距离曲线上下均为 *E*(*i*)长度。

* errorbar(X,Y,E) 用 ***X*** 和 ***Y*** 绘误差条图，误差条的长度为 2*E(i)。***X***、***Y*** 和 ***E*** 必须大小相同。当它们为向量时，每个误差条定义为(***X***(*i*),***Y***(*i*))处距离曲线上下各 ***E***(*i*)长度的误差条。当它们为矩阵时，每个误差条由(***X***(*i*,*j*),***Y***(*i*,*j*))定义。
* errorbar(X,Y,L,U) 用 ***L***(*i*)+***U***(*i*)指定误差条上下的长度，绘制误差条图。***X***,***Y***,***L*** 和 ***U*** 必须大小相同。当它们为向量时，每个误差条由(***X***(*i*),***Y***(*i*))定义，用 ***L***(*i*)定义曲线下方的距离，用 ***U***(*i*)定义曲线上方的距离。当它们为矩阵时，每个误差条由(***X***(*i*,*j*),***Y***(*i*,*j*))定义，用 ***L***(*i*,*j*)定义曲线下方的距离，用 ***U***(*i*,*j*)定义曲线上方的距离。
* errorbar(...,LineSpec) 用 LineSpec 指定线型、标记和颜色绘制误差条图。
* h = errorbar(...) 返回直线图形对象的句柄向量。

下面用 errorbar 函数绘一个误差条图。

```
y=[10 6 17 13 20];
e=[2 1.5 1 3 1];
errorbar(y,e)
```

生成图 7-4。

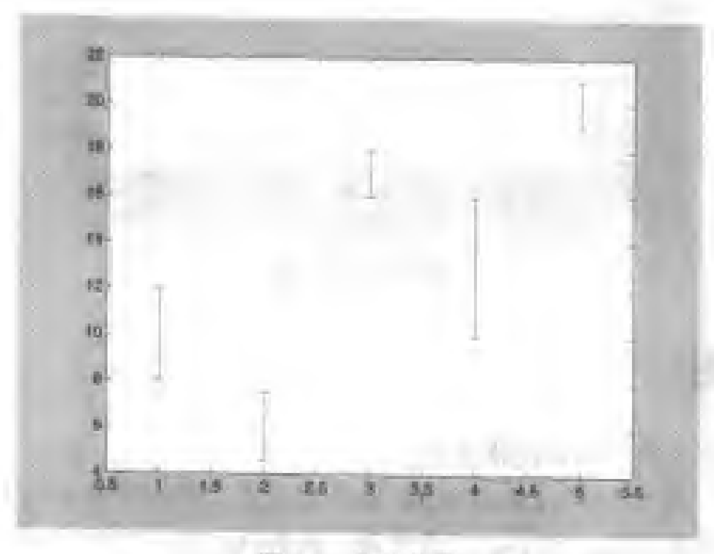


图 7-4 误差条图

## 7.4 散点图

分别用 scatter 函数和 scatter3 函数绘制二维和三维散点图。

scatter 函数的调用格式为：

* scatter(X,Y,S,C) 在向量 ***X*** 和 ***Y*** 指定的位置上显示彩色圆圈。***X*** 和 ***Y*** 必须大小相同。*S* 确定标记的大小。它可以是与 ***X*** 和 ***Y*** 长度相同的向量，也可以是一个标量。若 ***S*** 为标量，则 MATLAB 将所有的标记绘成相同大小。***C*** 确定每个标记的颜色，当 ***C*** 为与 ***X*** 和 ***Y*** 长度相同的向量时，将根据 ***C*** 中的值进行线性着色。当 ***C*** 为 length(X)×3 的矩阵时，用 RGB 值指定标记的颜色。***C*** 也可以是一个颜色字符串。
* scatter(X,Y) 用大小和颜色的默认设置绘标记。
* scatter(X,Y,S) 使用一种颜色，用指定的大小绘标记。

- scatter(...,markertype) 用指定的标记类型替代'o'。
- scatter(...,'filled') 填充标记。
- h = scatter(...) 返回 scatter 函数创建的直线对象的句柄。

下面利用随机数绘散点图。

```
x=[1:40];
y=rand(size(x));
Scatter(x,y)
```

生成图 7-5。

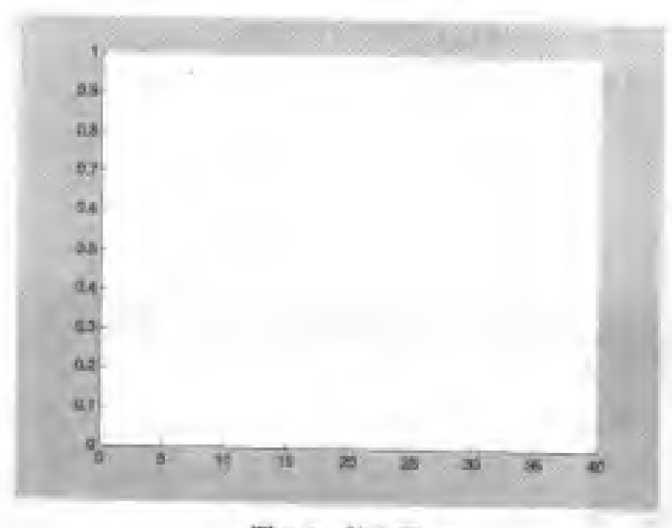


图 7-5　散点图

## 7.5　直方图

用 hist 函数绘直方图。其调用格式为：

- n = hist(Y)　将 *Y* 中的元素分成 10 份，然后用间隔相同的条形表示，返回每个条形中元素的个数。若 *Y* 是矩阵，则 hist 函数对每一列数据生成一个直方图。
- n = hist(Y,x)　其中 *x* 为向量，可以确定图中的条形数目。如，若 *x* 为 5 元素向量，则 hist 函数将 *Y* 中的元素分配到 5 组条形中。
- n = hist(Y,nbins)　其中 nbins 为标量，使用 nbins 组条形。
- [n,xout] = hist(...)　返回包含频数和条形位置的向量 ***n*** 和 ***xout***。可以使用 bar(xout,n) 来绘直方图。
- hist(...)　创建一个上面描述的直方图。hist 函数在 *Y* 的最小值和最大值之间沿 *x* 轴分配条形。
- hist(axes_handle,...)　将图形绘制到句柄 axes_handle 所表示的坐标系中。

hist 函数用等宽度的条形表示 *Y* 中数据的分布特征。如果 *Y* 是一个向量并且是惟一向量，则 hist 函数最多创建 10 个条形。例如，

```
yn = randn(10000,1);
hist(yn)
```

生成 10 000 个随机数，并沿 *x* 轴生成有 10 个条形的直方图，如图 7-6 所示。

*Y* 是矩阵时，hist 函数为每个列创建一组条形，用单独的颜色显示每组条形。例如，下面的语句为 *Y* 中的每列数据创建一组有 10 个条形的直方图。

```
Y = randn(10000,3);
hist(Y)
```

生成图 7-7。

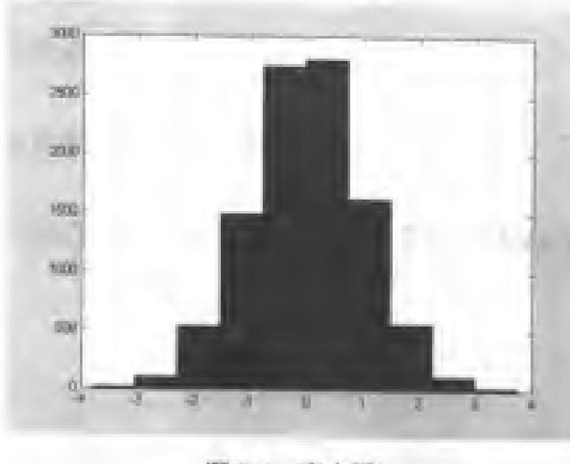


图 7-6 直方图

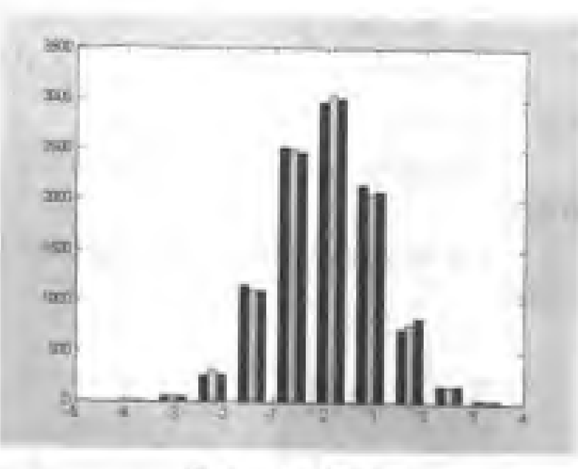


图 7-7 复合直方图

## 7.6 对数坐标图和半对数坐标图

在 MATLAB 中，用 loglog 函数可以实现双对数坐标转换，用 semilogx 和 semilogy 函数可以实现单轴对数坐标转换。

### 1. 对数坐标图

用 loglog 函数绘对数-对数比例图。其调用格式为：

- loglog(Y) 若 *Y* 的列值均为实数，则根据 *Y* 的列值和它们的对应编号绘图。若 *Y* 的列值为复数，则 loglog(Y)和 loglog(real(Y),imag(Y))等价，即根据 *Y* 各元素的实部和虚部数据绘图。

- loglog(X1,Y1,...) 根据 ***Xn*** 和 ***Yn*** 匹配数据绘图。若 ***Xn*** 和 ***Yn*** 中只有一个为矩阵，则 loglog 函数绘制向量变量与矩阵行或列的配套数据的图，它取决于向量的行或列的维数是否与矩阵配套。

- loglog(X1,Y1,LineSpec,...) 绘制所有由 ***Xn***,***Yn*** 和 LineSpec 等定义的线条。其中，LineSpec 决定线型、标记和图中直线的颜色。

- loglog(...,'PropertyName',PropertyValue,...) 给 loglog 函数创建的所有直线对象设置属性值。

- h = loglog(...) 返回句柄列向量到直线图形对象，一个句柄对应一条直线。

### 2. 半对数坐标图

用 semilogx 函数和 semilogy 函数分别对 *x* 轴和 *y* 轴绘半对数坐标数据图。其调用格式为：

- semilogx(Y) 令 *x* 轴取以 10 为底的对数比例，*y* 轴取线性比例。如果 *Y* 的值为实数，则根据 *Y* 的列值和它们对应的编号绘图。如果 *Y* 的值为复数，则 semilogx(Y)函数等价

于 semilogx(real(Y), imag(Y))。

- semilogx(X1,Y1,...) 根据所有的 ***Xn*** 和 ***Yn*** 配对数据绘图。如果 ***Xn*** 和 ***Yn*** 中只有一个为矩阵，则 semilogx 函数绘向量变量与矩阵的行或列的数据图。取行还是取列决定于是向量的行还是向量的列的维数与矩阵相匹配。
- semilogx(X1,Y1,LineSpec,...) 绘制所有由 ***Xn***,***Yn***,LineSpec 等定义的直线。LineSpec 确定线型、标记和线条的颜色。
- semilogx(...,'PropertyName',PropertyValue,...) 为所有由 semilogx 函数创建的直线图形对象设置属性值。
- semilogy(...) 用以 10 为底的对数比例定义 $y$ 轴，$x$ 轴取线性比例，在该坐标系中绘数据图。
- h = semilogx(...)和 h = semilogy(...) 返回直线图形对象的句柄向量，一条直线对应一个句柄。

## 7.7 多轴图

在同一个图中绘制坐标度量单位不同的图形，可以使图形表达更加简练。在有些情况下，多轴图有利于数据对比。

利用 plotyy 函数可以绘制双轴图。其调用格式为：

- plotyy(X1,Y1,X2,Y2) 用标注在图形左侧的 $y$ 轴单位绘 **X1** 和 **Y1** 的图形，用标注在图形右侧的 $y$ 轴单位绘 **X2** 和 **Y2** 的图形。
- plotyy(X1,Y1,X2,Y2,'function') 用字符串“function”指定的函数绘制每个图形。“function”可以是 plot,semilogx,semilogy,loglog,stem 或所有接受下面语法的 MATLAB 函数：

```
h = function(x,y)
```

- plotyy(X1,Y1,X2,Y2,'function1','function2') 对于图形左侧的坐标轴用 function1(X1,Y1) 绘数据图，对于右侧的坐标轴用 function2(X2,Y2)绘数据图。
- [AX,H1,H2] = plotyy(...) 返回 ***AX*** 中创建的两个坐标轴的句柄以及 *H*1 和 *H*2 中每个图形绘图对象的句柄。***AX***(1)为左侧轴，***AX***(2)为右侧轴。

利用 plotyy 命令可以创建两套数据的图形，并同时使用左侧和右侧的 $y$ 轴。对于每套数据，还可以应用不同的绘图函数。例如，可以在同一幅图中组合线形图和火柴杆图。

```
t = 0:pi/20:2*pi;
y = exp(sin(t));
plotyy(t,y,t,y,'plot','stem')
```

结果如图 7-8 所示。

可以用 plotyy 函数在同一幅图中应用线性坐标和对数坐标。

```
t = 0:900; A = 1000; a = 0.005; b = 0.005;
z1 = A*exp(-a*t);
z2 = sin(b*t);
[haxes,hline1,hline2] = plotyy(t,z1,t,z2,'semilogy','plot');
axes(haxes(1))
```

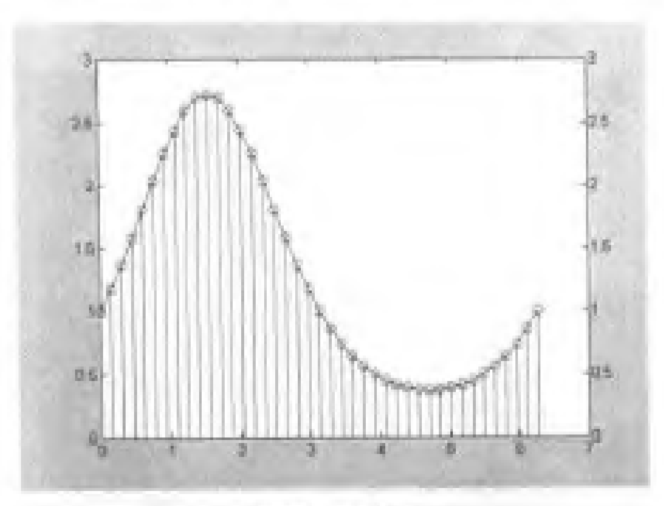


图 7-8　双轴图

```
ylabel('Semilog Plot')
axes(haxes(2))
ylabel('Linear Plot')
set(hline2,'LineStyle','--')
```

生成图 7-9。

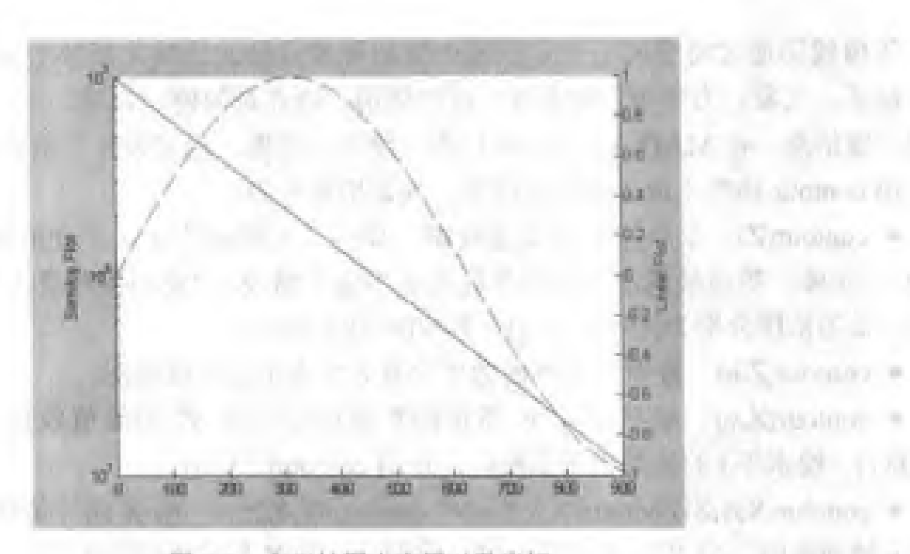


图 7-9　在双轴图中应用对数坐标

## 7.8　极坐标图

用 polar 函数绘极坐标图。其调用格式为：

- polar(theta,rho)　根据角度 *theta* 和半径 *rho* 创建极坐标图。
- polar(theta,rho,LineSpec)　LineSpec 指定极坐标图中直线的线型、标记和颜色。

下面绘制一个简单的极坐标图。

```
t = 0:.01:2*pi;
polar(t,sin(2*t).*sin(2*t),'--r')
```

如图 7-10 所示。

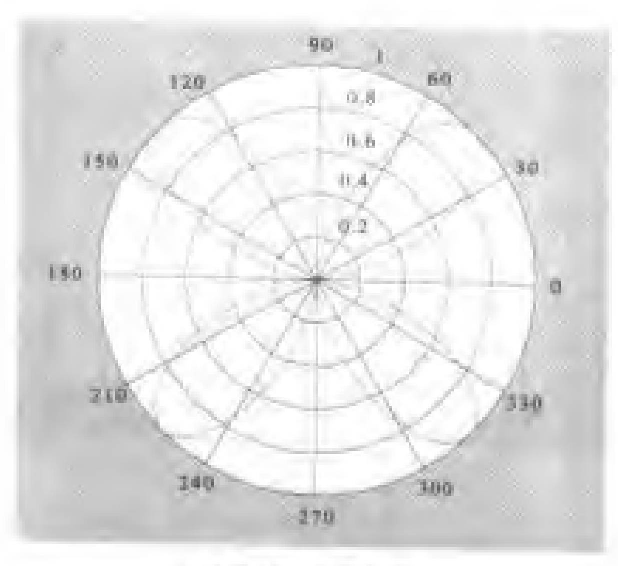



图 7-10　极坐标图

## 7.9　等值线图

等值线图通过将空间上一定范围内值相等的点依次连线来反映数据的分布特征。这种图在地质、气象、力学等领域有着广泛的应用，特点是简便、直观，而且二维等值线图还能表现三维信息。在 MATLAB 中，可以作二维等值线图，也可以作三维等值线图。

用 contour 函数生成二维等值线图。其调用格式为：

- contour(Z)　绘矩阵 **Z** 的等值线图，其中 **Z** 可解释为 $x$-$y$ 平面的高度。**Z** 必须至少是 2×2 的矩阵。等值线的水平数和等值线水平值根据 **Z** 的最小值和最大值自动进行选取。$x$ 轴和 $y$ 轴的范围分别为[1:n]和[1:m]，其中[m,n] = size(Z)。
- contour(Z,n)　根据 Z 矩阵的数据绘有 $n$ 个水平的等值线图。
- contour(Z,v)　根据向量 $v$ 指定的数据值绘矩阵 **Z** 的等值线图。等值线水平等于 length(v)。绘水平 $i$ 的的单条等值线时，使用 contour(Z,[i i])。
- contour(X,Y,Z),contour(X,Y,Z,n)和 contour(X,Y,Z,v)　绘 **Z** 的等值线图。***X*** 和 ***Y*** 指定 $x$ 轴和 $y$ 轴的范围。当 ***X*** 和 ***Y*** 为矩阵时，它们必须与 **Z** 具有相同的大小。
- contour(...,LineSpec)　用 LineSpec 指定的线型和颜色绘等值线。contour 函数忽略标记。
- [C,h] = contour(...)　返回等值线矩阵 ***C*** 和图形对象的句柄向量。

利用 peaks 数据，在命令窗口键入

```
contour(peaks(50),15);
colormap cool
```

得等值线图如图 7-11 所示。

使用 interp2 函数和 contour 函数创建平滑化的等值线图。

```
P = Peaks(50);
contour(interp2(P,5));
```

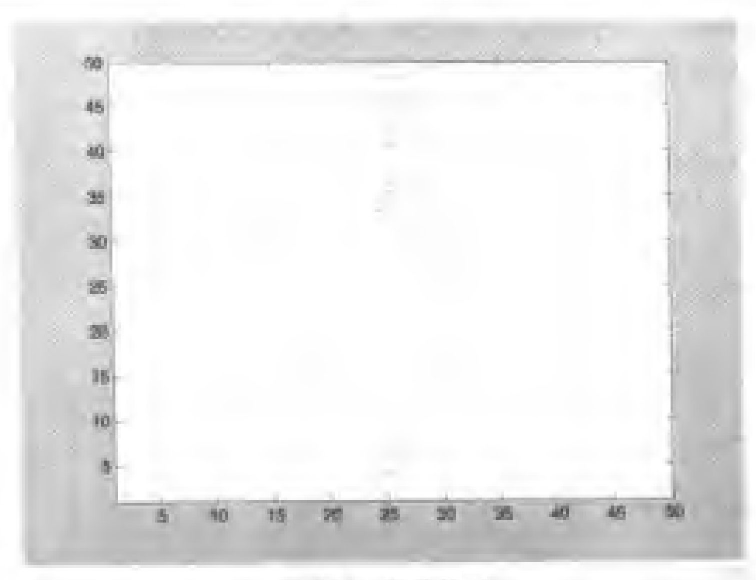



图 7-11　等值线图

生成图 7-12。进行平滑化需要一定的计算时间。

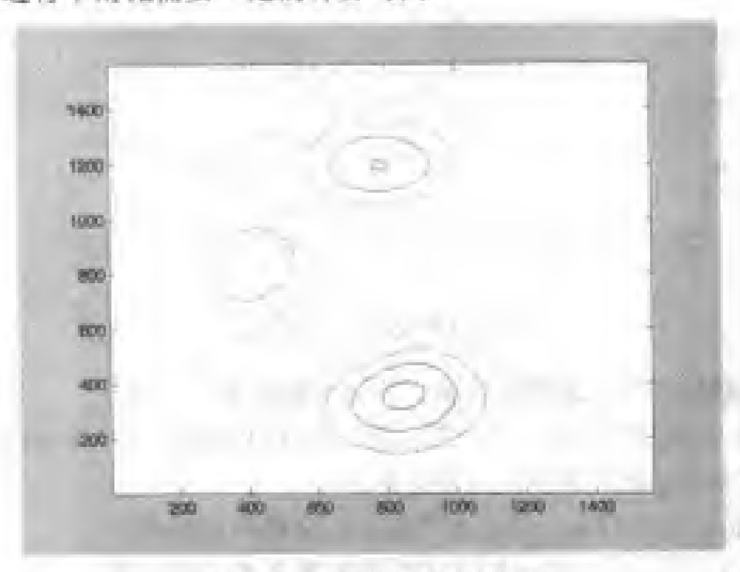



图 7-12　平滑化后的等值线图

用 clabel 函数在等值线图中进行标注。其调用格式为：

- clabel(C,h)　旋转标签并将它们插到等值线中。该函数只插入那些在等值线图中合适的标签，它决定于等值线图的大小。
- clabel(C,h,v)　*v* 向量给定等值线水平，创建标签，然后旋转标签并将它们插到等值线中去。
- clabel(C,h,'manual')　将等值线标签放到鼠标选定的位置。单击鼠标左键，在最靠近图标中心位置下方的位置上进行标注。当图标处于图形窗口中时，单击回车键终止标注。旋转标签并插入到等值线图中。
- clabel(C)　根据等值线结构参数 ***C***（contour 函数的输出）的值把标签添加到当前等值线图中。该函数标注所有显示的等值线并随机选择标注位置。
- clabel(C,v)　只标注向量 *v* 中给定的等值线水平。

- clabel(C,'manual')　在鼠标选定的位置上放置等值线标签。

下面创建、绘制并标注简单等值线图。

```
Z = peaks;
[C,h] = contour(interp2(Z,4));
text_handle = clabel(C,h);
set(text_handle,'BackgroundColor',[1 1 .6],...
    'Edgecolor',[.7 .7 .7])
```

生成图 7-13。

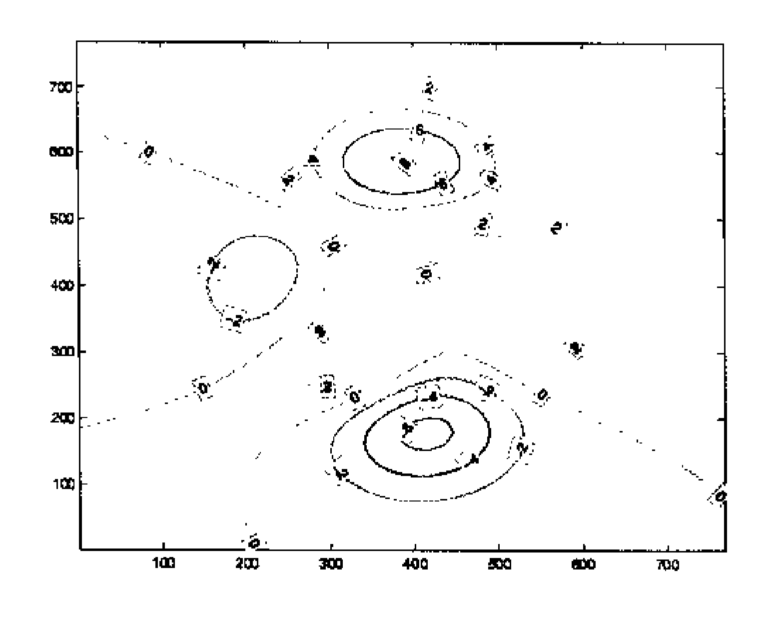



图 7-13　等值线图的标注

用 contourf 函数填充的二维等值线图。其调用格式为：

- contourf(Z)　绘矩阵 **Z** 的等值线图，其中 **Z** 为平面的高度。**Z** 必须至少为 2×2 的矩阵。等值线的个数和等值线对应的值自动选择确定。
- contourf(Z,n)　绘一有 *n* 个等值水平的矩阵 **Z** 的等值线图。
- contourf(Z,v)　绘向量 *v* 指定的水平数的矩阵 **Z** 的等值线图。
- contourf(X,Y,Z), contourf(X,Y,Z,n)和 contourf(X,Y,Z,v)　用 ***X*** 和 ***Y*** 确定 *x* 轴和 *y* 轴的范围，生成 **Z** 的等值线图。当 ***X*** 和 ***Y*** 为矩阵时，它们必须与 **Z** 的大小相同。
- [C,h,CF] = contourf(...)　返回等值线矩阵 ***C***、句柄向量 *h* 和填充区域的等值线矩阵 ***CF***。

下面创建 peaks 函数的填充等值线图。

```
Contourf(peaks(40),15);
colormap jet
```

生成图 7-14。

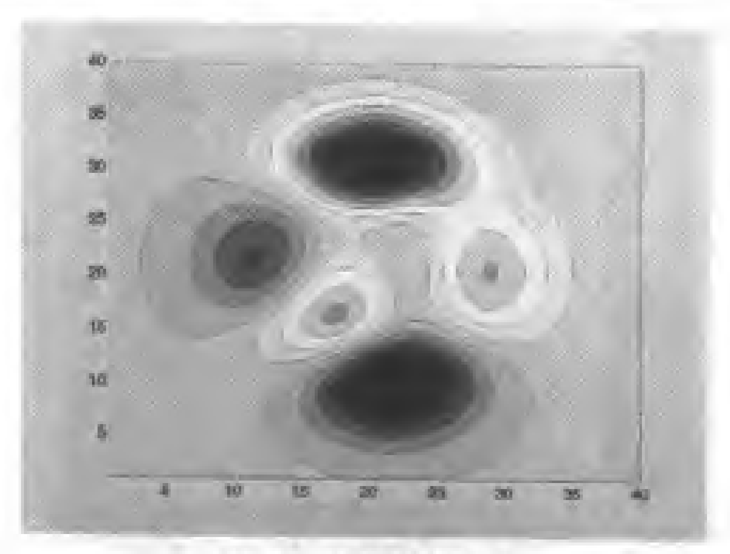


图 7-14　填充的等值线图

## 7.10　向量图

向量图用箭头显示图中各点处的向量大小和方向。其中，箭头指示的方向为向量的方向，箭头的长短表示向量的大小。

用 quiver 函数向量或速率图。其调用格式为：

- quiver(U,V)　在由 $x = 1:n$ 和 $y = 1:m$ 确定的坐标系中绘 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 指定的向量图，其中 [m,n] = size(U) = size(V)。
- quiver(X,Y,U,V)　对每个 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 配对数据绘向量图。$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 为向量，length(X) $=n$ 且 length(Y) $= m$，其中[m,n] = size(U) = size(V)。向量 $\boldsymbol{X}$ 对应于 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 的列，向量 $\boldsymbol{Y}$ 对应于 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 的行。
- quiver(...,scale)　自动对向量设置显示比例。如果 scale=2，则将原向量的长度乘以 2 以后显示。若 scale=0，则不自动设置显示比例。
- quiver(...,LineSpec)　用 LineSpec 参数指定线型、标注和颜色。quiver 函数在向量原点处绘标记。
- quiver(...,LineSpec,'filled')　填充由 LineSpec 指定的标记。
- h = quiver(...)　返回直线句柄向量。

下面绘函数的梯度场

```
[x,y,z]= peaks(30);
[dx,dy] = gradient(z,.2,.2);
contour(x,y,z)
hold on
quiver(x,y,dx,dy)
colormap autumn
grid off
hold off
```

绘图结果如图 7-15 所示。

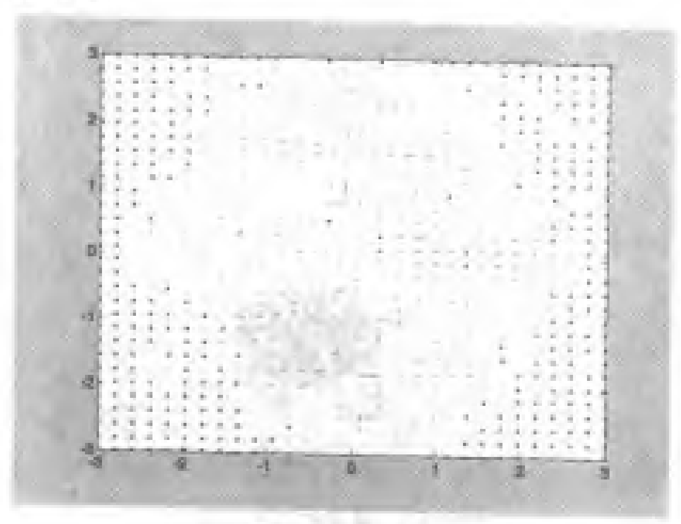


图 7-15　向量图

## 7.11　帕累托图

帕累托图按降序用条形表示向量中的值。用 pareto 函数绘制帕累托图，该函数的语法格式为：

- pareto(Y)　用 *Y* 中的元素编号标注每个条形。
- pareto(Y,names)　用字符串矩阵或单元数组 *names* 中的相关名称标注每个条形。
- pareto(Y,X)　用 *X* 中的相关值标注每个条形。
- H=pareto(...)　返回 patch 和 line 对象句柄的组合。

假设下面 *y* 中的数据表示各元素占总体的百分数，绘帕累托图。

```
y=[8 9 50 20 3 10];
pareto(y)
```

生成图 7-16。

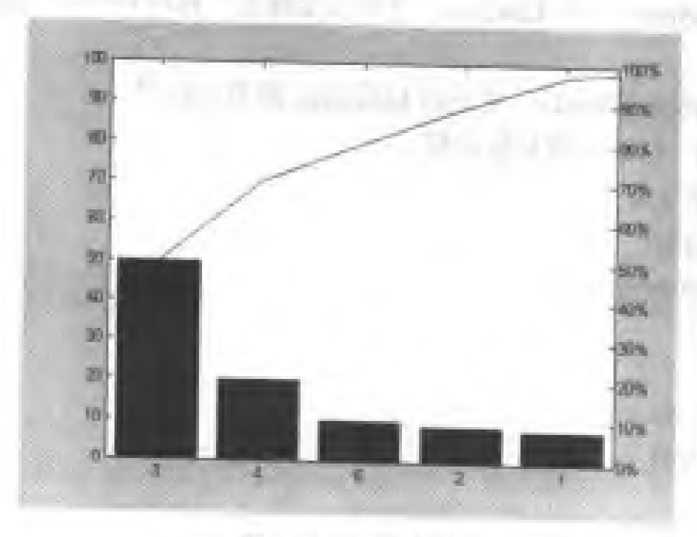


图 7-16　帕累托图

## 7.12　火柴杆图

火柴杆图沿 $x$ 轴将数据用直线段相对于基线显示在上下两侧，数据点用小圆圈或其他标记显示。

用 stem 函数绘制火柴杆图，该函数的语法格式如下所示。

- stem(Y)　将 **Y** 中的数据沿 $x$ 轴用直线段相对于基线等间隔排列。如果 **Y** 是矩阵，则 stem 函数对应于同一个 $x$ 值绘制行中的所有元素。
- stem(X,Y)　绘 **X** 和 **Y** 的列数据的图形。**X** 和 **Y** 必须是大小相同的向量或矩阵。另外，**X** 可以是一个行向量或列向量，**Y** 可以是一个有 length(X)行的矩阵。
- stem(..., 'fill')　指定是否对火柴杆末端的圆圈着色。
- stem(...,LineSpec)　指定火柴杆和末端标记的线型、标记类型和颜色。
- stem(axes_handles,...)　将图形绘制到 axes_handles 表示的坐标系中。
- h=stem(...)　将一个 stemseries 对象句柄向量返回到 $h$ 中，**Y** 中每列数据有一个句柄。

下面是创建一幅火柴杆图的例子，表示 0 和 2π 之间线性间隔的 10 个值的余弦值。注意，设置基线的线型时，首先用 stemseries 对象的 BaseLine 属性获取基线的句柄。

```
t = linspace(-2*pi,2*pi,10);
h = stem(t,cos(t),'fill','--');
set(get(h,'BaseLine'),'LineStyle',':')
set(h,'MarkerFaceColor','red')
```

生成图 7-17。

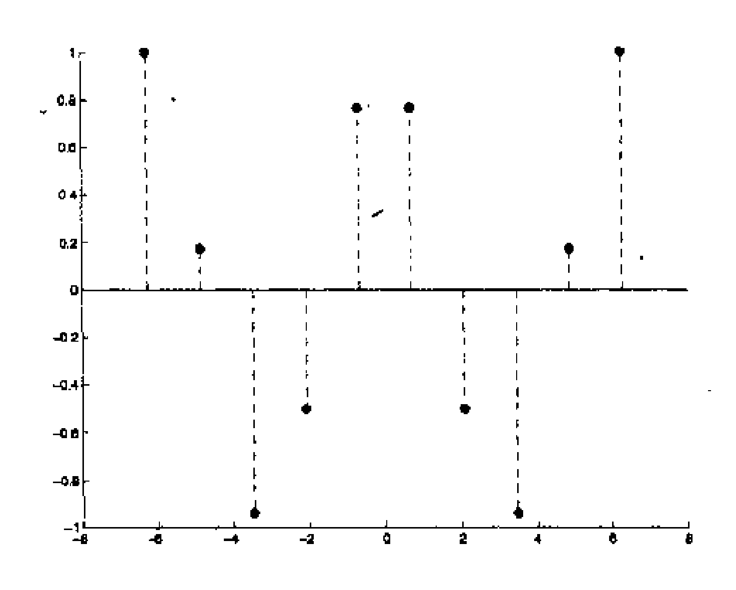



图 7-17　火柴杆图

下面用一个两列矩阵创建火柴杆图。此时，stem 函数创建两个 stemseries 对象，一列数据对应一个对象。这两个对象的句柄都返回输出变量 $h$ 中。

```
x = 0:25;
y = [exp(-.07*x).*cos(x);exp(.05*x).*cos(x)]';
h = stem(x,y);
set(h(1),'MarkerFaceColor','blue')
set(h(2),'MarkerFaceColor','red','Marker','square')
```

生成图 7-18。

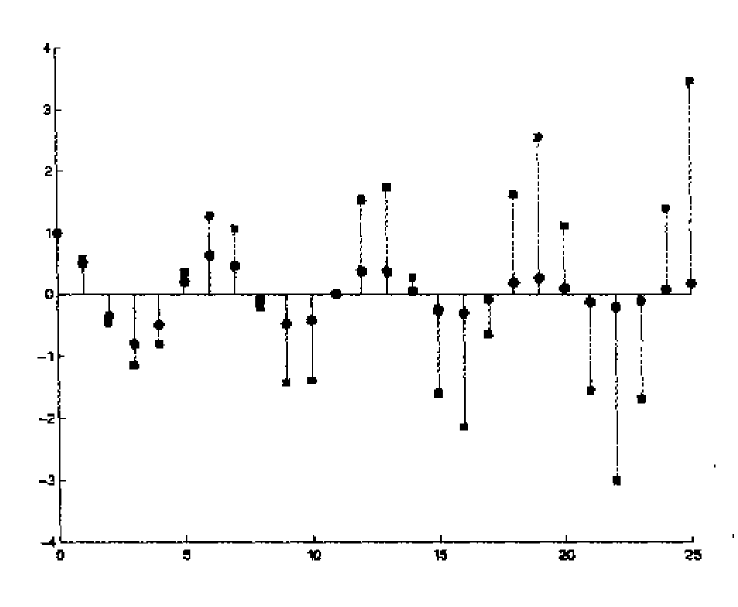


图 7-18　复合火柴杆图

## 7.13　彗星图

彗星图实际上是一个动画，用一个小圆圈(彗星头)跟踪屏幕上的数据点。彗星体是头后面的跟踪线段，彗星尾是跟踪整个函数的实线。

用 comet 函数绘制彗星图，该函数的语法格式为：

- comet(y)　显示向量 ***y*** 的彗星图。
- comet(x,y)　显示向量 ***y*** 和向量 ***x*** 的彗星图。
- comet(x,y,p)　指定长度为 p*length(y)的彗星体，默认时 p=0.1。
- comet(axes_handle,...)　将图形绘制到 axes_handle 指定的坐标系中。

下面的例子创建一个简单的彗星图。

```
t = 0:.01:2*pi;
x = cos(2*t).*(cos(t).^2);
y = sin(2*t).*(sin(t).^2);
comet(x,y);
```

生成图 7-19。

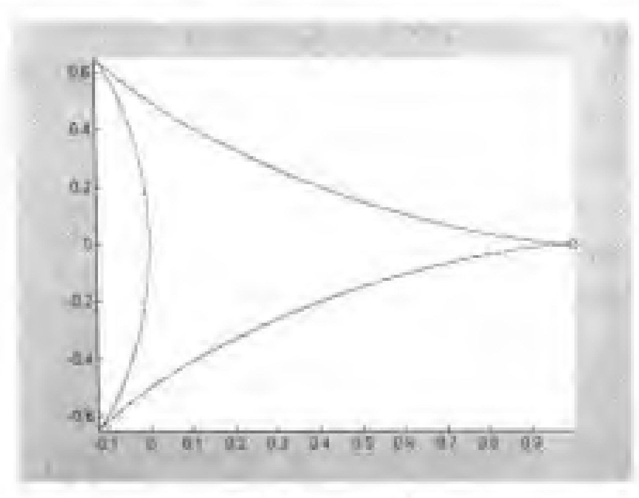


图 7-19　彗星图

## 7.14　罗盘图

罗盘图用起点在原点的箭头表示向量数据，数据为笛卡儿坐标中的值，显示在圆形网格中。

用 compass 函数绘罗盘图，该函数的语法格式为：

- compass(U,V)　显示有 *n* 个箭头的罗盘图，其中 *n* 是 ***U*** 或 ***V*** 中的元素个数。每个箭头的起点在原点。每个箭头的端点为相对于原点，由[***U***(*i*),***V***(*i*)]确定的点。
- compass(Z)　显示一个有 *n* 个箭头的罗盘图，其中 *n* 是 **Z** 中的元素个数。
- compass(…,LineSpec)　用 LineSpec 指定的线型、标记和颜色绘图。
- compass(axes_handle,…)　将图形绘制在 axes_handle 指定的坐标系中。
- h=compass(…)　将句柄返回 Line 对象。

下面的例子是绘制一个矩阵特征值的罗盘图。

```
Z = eig(randn(20 , 20));
compass(Z)
```

生成图 7-20。

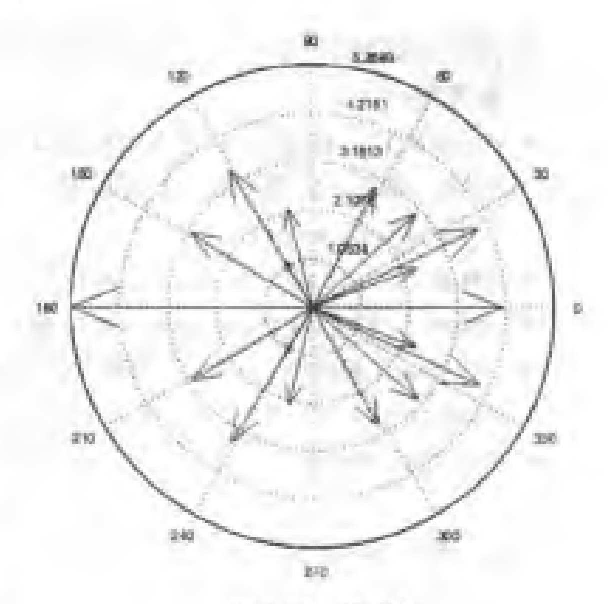


图 7-20　罗盘图

## 7.15　羽列图

羽列图沿水平轴在等间隔的点上显示向量，用 feather 函数绘制羽列图，该函数的语法格式为：

- feather(U,V)　显示 ***U*** 和 ***V*** 指定的向量，其中 ***U*** 和 ***V*** 分别包含作为相对坐标 *x* 和 y 组分的数据。
- feather(Z)　显示 **Z** 中复数值指定的向量，等价于 feather(real(Z),imag(Z))。

- feather(...,lineSpec)　用 LineSpec 指定的线型、标记和颜色绘羽列图。
- feather(axes_handle,...)　将图形绘制到 axes_handle 指定的坐标系中。
- h=feather(...)　将直线对象的句柄返回 *h* 中。

下面的例子用一幅羽列图显示 theta 向量的方向。

```
theta = (-90:10:90)*pi/180;
r = 2*ones(size(theta));
[u,v] = pol2cart(theta,r);
feather(u,v);
```

生成图 7-21。

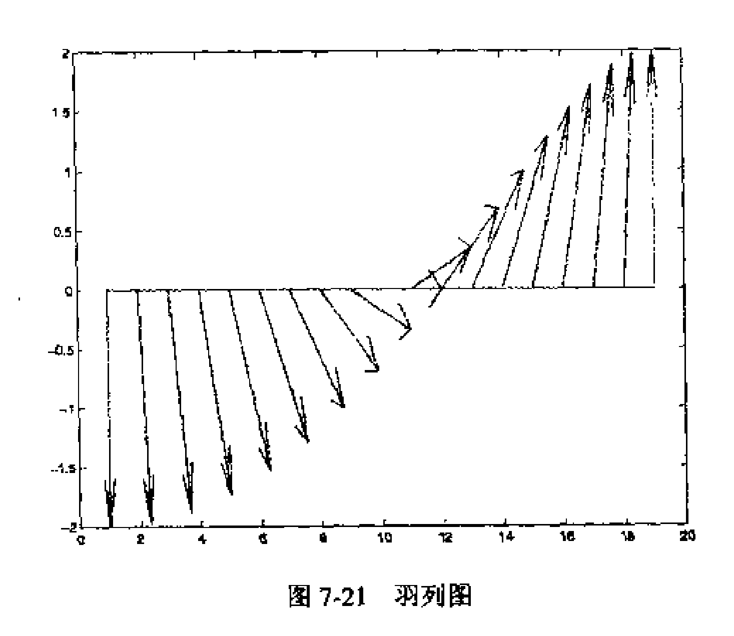



图 7-21　羽列图

## 7.16　阶梯图

阶梯图主要用于绘制数字采样数据的时间历史图形。用 stairs 函数绘制阶梯图，该函数的语法格式为：

- stairs(X,Y)　在 *Z* 指定的点上绘制 *Y* 的元素。*X* 的元素必须是单调的。*X* 的大小必须与 *Y* 的相等，或者，*Y* 是矩阵，*X* 可以是一个行向量或列向量，使得

```
length(X) = size(Y,1)
```

- stairs(...,LineSpec)　用 LineSpec 指定的线型、标记和颜色绘图。
- stairs(..., 'PropertyName',propertyvalue)　采用指定的属性设置绘图。
- stairs(axes_handles,...)　将图形绘制到 axes_handles 指定的坐标系中。
- h=stairs(...)　返回 stairseries 对象创建的句柄。
- [xb,yb]=stairs(Y,...)　不绘图，但是返回向量 ***xb*** 和 ***yb***，这样，可以用 plot(xb,yb)绘阶梯图。

下面的例子是绘制正弦波的阶梯图。

```
x = linspace(-2*pi,2*pi,40);
stairs(x,sin(x))
```

生成图 7-22。

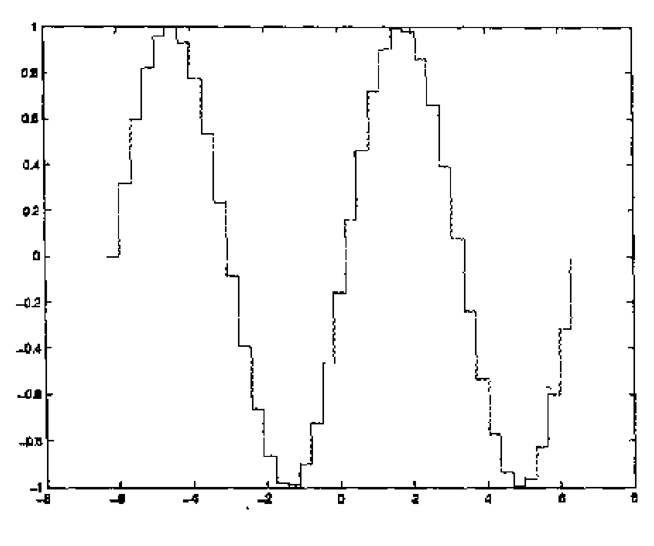


图 7-22　正弦波的阶梯图

## 7.17　玫瑰花图

玫瑰花图实际上是极坐标中的直方图，它根据数据值的范围进行分组，并显示数据的分布特征。每个组显示为一瓣。

用 rose 函数绘制玫瑰花图，该函数的语法格式为：

- rose(theta)　用 *theta* 数据绘制玫瑰花图，玫瑰花的瓣数一般为 20，或更少。向量 *theta* 中的数据用弧度表示，确定每个花瓣的角度。花瓣的长度反映了 *theta* 中落在该范围内的元素的个数。
- rose(theta,x)　用向量 ***x*** 指定瓣的个数和位置。length(x)为瓣数，***x*** 的值指定每个瓣的角度。
- rose(theta,nbins)　在[0,2*pi]范围内绘制 nbins 个等间隔的瓣。默认值为 2.0。
- rose(axes_handle,...)　将图形绘制到 axes_handle 指定的坐标系中。
- h=rose(...)　返回向量 ***tout*** 和 ***rout***，这样，可以用 polar(tout,rout)绘图。

下面的例子用玫瑰花图显示 50 个随机数的分布特征。

```
theta = 2*pi*rand(1,50);
rose(theta)
```

生成图 7-23。

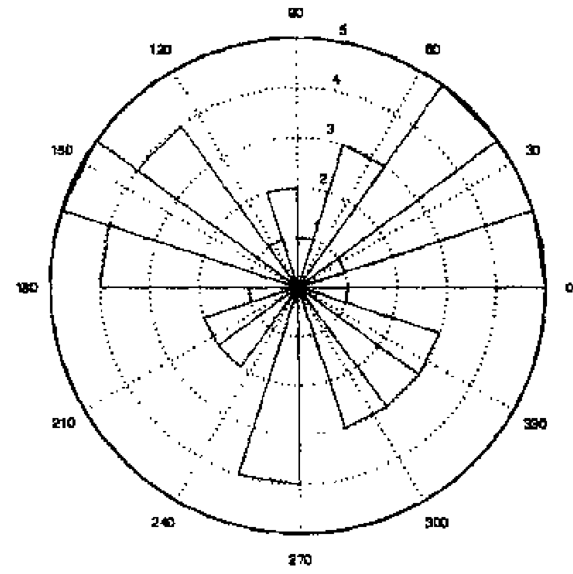



图 7-23　玫瑰花图

## 7.18 函数的图形

可以用 fplot 函数在指定范围内绘函数的图形。函数必须是 $y=f(x)$形式的，其中，$x$ 是向量，为自变量；$y$ 为向量，为因变量。该函数的语法格式为：

- fplot(function,limits)　在 **limits** 指定的范围内绘制 function 函数的图形。**limits** 是一个向量，指定 $x$ 轴上的范围[xmin xmax]，或者 $x$ 轴和 $y$ 轴上的范围[xmin xmax ymin ymax]。function 必须是 M 文件函数的名称或句柄，或者含有变量 $x$ 的字符串。
- fplot(function,limits,LineSpec)　用 LineSpec 指定的属性绘图。
- fplot(function,limits,tol)　用相对容限 tol 绘图，默认时，tol 为 2e-3。
- fplot(function,limits,tol,LineSpec)　使用相对误差容限 tol 和 LineSpec 指定的属性绘图。
- fplot(function,limits,n)　用最少 $n$+1 个点绘函数的图形。默认时 $n$ 为 1。最大步长限制为(1/$n$)*($x$max-$x$min)。
- fplot(fun,lims,…)　接受可选变量 tol,n 和 LineSpec 的组合。
- fplot(axes_handle,...)　将图形绘制到 axes_handle 指定的坐标系中。
- [X,Y]=fplot(function,limits,...)　将横坐标和纵坐标返回到 $X$ 和 $Y$ 中。该格式不绘图，但可以用 plot(X,Y)绘图。
- [...]=fplot(function,limits,tol,n,LineSpec,P1,P2,...)　使用此格式，可以直接将参数 P1,P2 等传递给函数。
- Y=function(X,P1,P2,...)

传递空矩阵([ ])，可以使用 tol,n 或 lineSpec 的默认值。

下面的例子是在[-2 2]范围内绘双曲正切线。

```
fplot('tanh',[-2 2])
```

生成图 7-24。

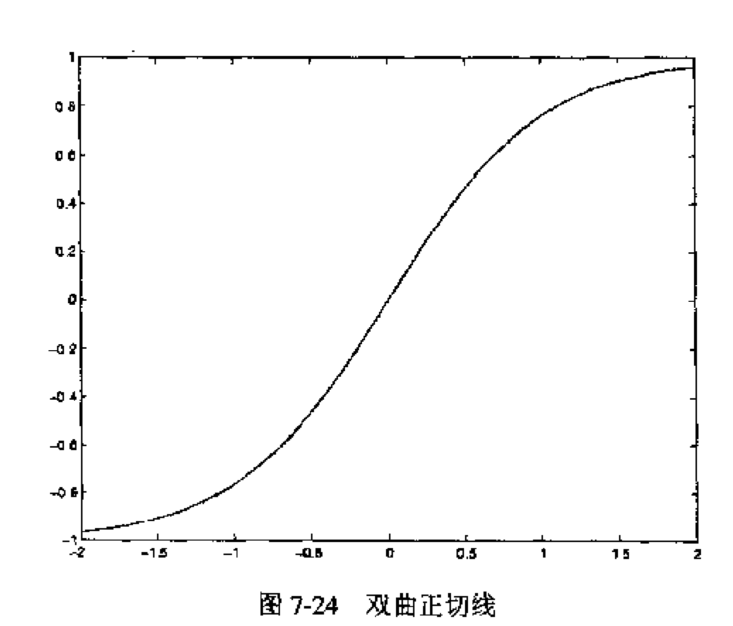



图 7-24　双曲正切线

创建一个 M 文件 myfun，它返回一个两列矩阵。

```
function Y = myfun(x)
Y(:,1) = 200*sin(x(:))./x(:);
Y(:,2) = x(:).^2;
```

创建一个指向 myfun 的函数句柄。

```
fh = @myfun;
```

用下面的语句绘图。

```
fplot(fh,[-20 20])
```

生成图 7-25。

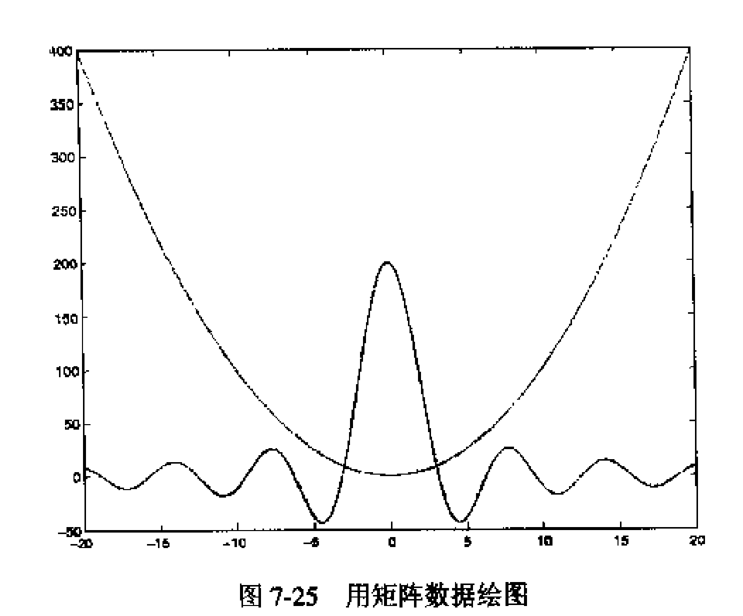



图 7-25　用矩阵数据绘图

## 7.19　动画

在 MATLAB 中，可以用两种方法创建动画序列：

- 保存很多不同的图片，然后以电影的形式进行显示；
- 在屏幕上连续擦除和重画对象。每次重画都作递增式的改变。

动画适合于每一帧都相当复杂，不能快速重画的情况。可以先创建每一个动画帧，回放时与原来绘制动画帧的时间没有关系。动画不是实时绘制的，它只是预先绘好的一系列帧的回放。

动画的第二种技巧，即绘制、擦除，然后重绘，确保 MATLAB 支持不同的绘图模式。这些模式允许在损失部分绘制精度的情况下，进行快速绘制。必须想好选择哪种模式。

### 7.19.1　以电影方式创建动画

可以保存任何图形序列，然后回放它。这里有两个步骤：

- 用 getframe 获取每个动画帧；
- 用 movie 命令运行动画。

一般地，在一个 for 循环中使用 getframe 命令组合动画帧的数组。getframe 命令返回一个具有下列字段的结构。

- cdata　以 uint8 型矩阵保存的图像数据。在索引颜色系统上，矩阵具有 height×width 维；在真彩色系统上，具有 height×width×3 维。
- colormap　是一个 $n×3$ 的矩阵，其中 $n$ 是颜色个数。在真彩色系统上，colormap 字段是空的。

下面演示如何用动画可视化 fft(eye(n))，它是一个复杂的 $n×n$ 矩阵。

**1．创建动画**

下面的代码在一个 for 循环中调用 getframe 函数捕获图形。因为 plot 命令重置坐标轴属性，在使用 getframe 前调用 axis equal 语句。

```
for k = 1:16
    plot(fft(eye(k+16)))
    axis equal
    M(k) = getframe;
end
```

生成图 7-26。

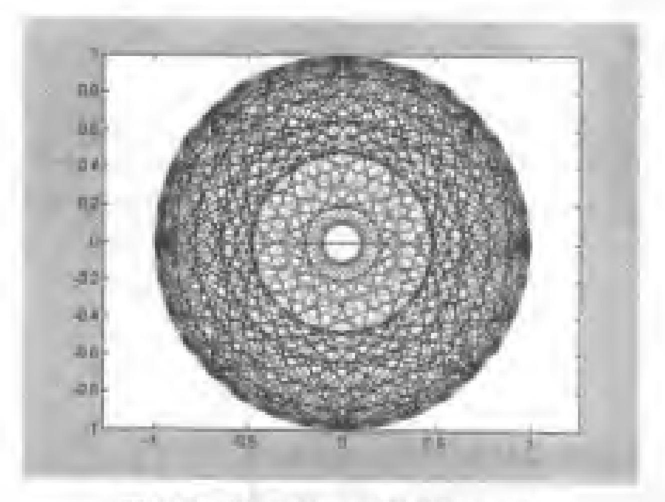


图 7-26　用 getframe 函数获取动画帧

**2．运行动画**

生成动画以后，可以将它回放任意次。例如回放 30 次，键入

```
movie(M,30)
```

在大部分计算机上，可以有序地创建和平稳回访数十帧动画。更多帧的动画需要计算机具有更大的内存或更有效的虚拟系统。

如果试图捕获整个图形窗口的内容，把图形句柄作为变量指定给 getframe 命令。例如，添加一个滚动条来指示上例中的 $k$ 值。

```
h = uicontrol('style','slider','position',...
    [10 50 20 300],'Min',1,'Max',16,'Value',1)
for k = 1:16
    plot(fft(eye(k+16)))
```

```
        axis equal
        set(h,'Value',k)
        M(k) = getframe(gcf);
    end
```

本例中，动画帧包括整幅图形。下面使回放坐标系充满整个图形窗口。

```
    clf
    axes('Position',[0 0 1 1])
    movie(M,30)
```

### 7.19.2　以重绘方式创建动画

可以通过重绘的方法绘制图形对象。不断重复绘制、擦除、重绘的过程，可以创建动画效果。进行重绘，需要首先选择擦除模式。擦除模式不同，会有不同的绘图效果。一共有三种绘图模式。

- none　对象移动时不擦除对象。
- background　通过用背景色重绘对象来达到擦除图形的目的。该模式会擦除对象和它下面的任何图形（如网格线等）。
- xor　该模式只擦除对象，通常用于动画。

下面的例子用名为 Lorenz 奇异吸引子的非线性差分方程描述无序运动。该方程具有下面的形式

$$\frac{dy}{dt} = Ay$$

$y(t)$为向量值函数，矩阵 $A$ 由 $y$ 确定，即

$$A(y) = \begin{bmatrix} -\frac{8}{3} & 0 & y(2) \\ 0 & -10 & 10 \\ -y(2) & 28 & -1 \end{bmatrix}$$

本例用最简单的欧拉法，采用固定步长进行求解。结果不是很精确，但是它与其他方法具有相同的定性行为。

```
A = [ -8/3 0 0; 0 -10 10; 0 28 -1 ];
y = [35 -10 -7]';
h = 0.01;
p = plot3(y(1),y(2),y(3),'.', ...
    'EraseMode','xor','MarkerSize',5); % 将 EraseMode 属性值设置为 none
axis([0 50 -25 25 -25 25])
hold on
for i=1:4000
    A(1,3) = y(2);
    A(3,1) = -y(2);
    ydot = A*y;
    y = y + h*ydot;
```

```
        % 改变坐标
        set(p,'XData',y(1),'YData',y(2),'ZData',y(3))
        drawnow
        i=i+1;
    end
```

生成图 7-27。

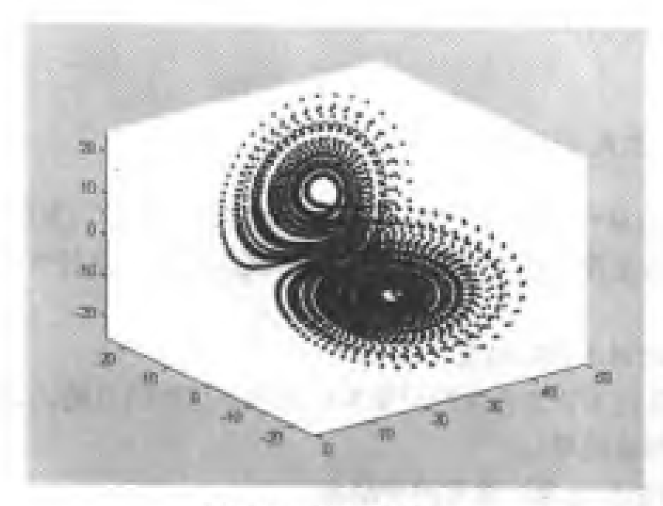


图 7-27　描述无序运动

在前面的程序后面添加下面的代码，可以查看擦除模式为 background 时的绘图效果。

```
    p = plot3(y(1),y(2),y(3),'square', ...
        'EraseMode','background','MarkerSize',10,...
        'MarkerEdgeColor',[1 .7 .7],'MarkerFaceColor',[1 .7 .7]);
    for i=1:4000
        A(1,3) = y(2);
        A(3,1) = -y(2);
        ydot = A*y;
        y = y + h*ydot;
        set(p,'XData',y(1),'YData',y(2),'ZData',y(3))
        drawnow
        i=i+1;
    end
    hold off
```

此时，会将原来的图形擦除。

将擦除模式设置为 xor 时，只能看到一个运动的点。

# 第 8 章　交互绘图与编辑

交互绘图和编辑是 MATLAB 7.0 的新特点。所谓交互，指的是绘图过程的可视性和可操作性更强：图形绘好以后，可以选择图中的图形元素，并针对选定的图形元素进行平移、拉伸、旋转、剪切和删除等操作。交互功能的增加，使 MATLAB 的图形创建和编辑工作更方便、更准确、更有趣了。

## 8.1　绘图工具

MATLAB 绘图函数和工具把图形绘制在一个单独的窗口中，这个窗口称为图形窗口，如图 8-1 所示。

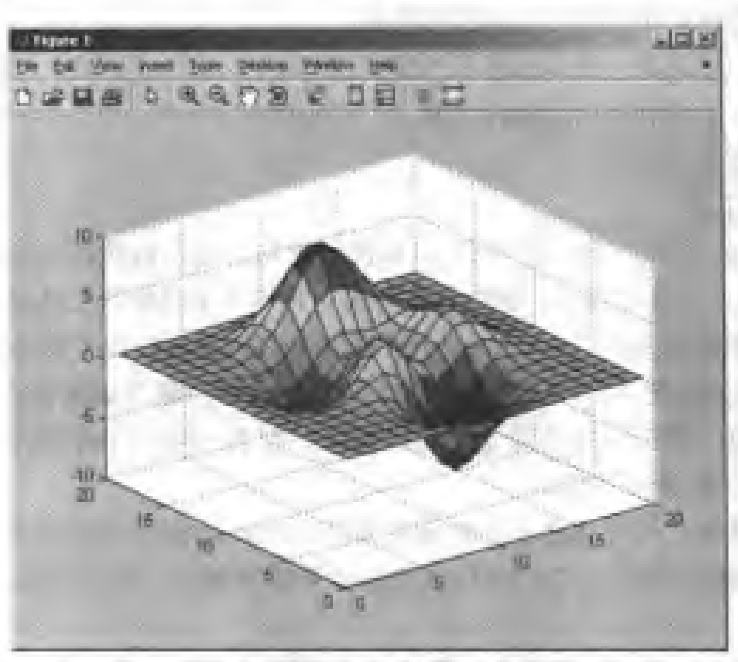


图 8-1　图形窗口

默认时，MATLAB 用线型和颜色区分图中的数据集，但是，可以改变这些图形组成元素的外观或在图中添加注释以说明数据的意义。

### 8.1.1　图形窗口的工具条

图形窗口中的工具条提供了使用常用功能的快捷方式，如图 8-2 所示。

注意，从“View”菜单中还可以启用另外两种工具条：

- 相机工具条　用于操作三维视图；
- 图形编辑工具条　用于标注和设置对象属性。

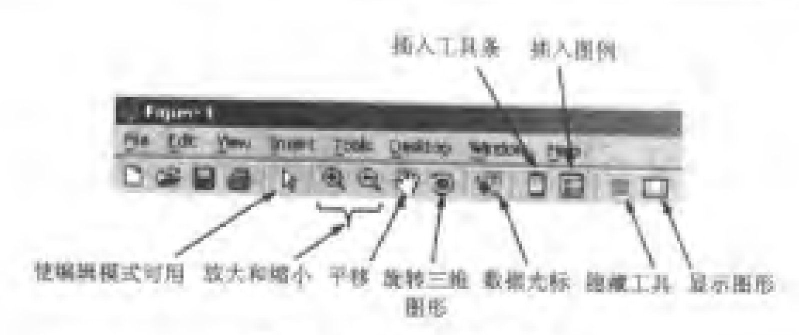



图 8-2　图形窗口的工具条

### 8.1.2　绘图工具——交互绘图

MATLAB 提供一组绘图工具集，它组成交互式的绘图环境，利用这个环境，可以完成以下任务：

- 创建不同类型的图形；
- 直接从工作空间浏览器中选择绘图变量；
- 在图形窗口中比较容易地创建和操作子图；
- 添加箭头、直线和文本等标注元素；
- 设置图形对象的属性。

**1．启动绘图工具**

用 plottools 命令可以创建带绘图工具的图形窗口，也可以通过单击▭图标来启动绘图工具。单击▭按钮，从图形窗口中删除绘图工具。图 8-3 示出绘图工具界面图。

在"View"菜单中选择"Figure Palatte"，"Plot Browser"或"Property Editor"选项，可以显示 3 种基本的绘图工具，即图形面板、绘图浏览器和属性编辑器。

（1）图形面板　用于创建和重置子图坐标系、察看和绘制工作空间变量并添加标注。用 figurepalette 命令显示图形面板。

（2）绘图浏览器　用于选择和控制图形窗口中坐标轴或图形对象的可见性。也可以通过单击"Add Button"按钮，给任何选定的坐标系添加数据。用 plotbrowser 命令显示绘图浏览器。

（3）属性编辑器　用于设置选定对象的公共属性。还可以单击"Inspector"按钮显示属性查看器，它提供获取所有对象属性的途径。用 propertyeditor 命令显示属性编辑器。

**2．图形面板**

图形面板包括 3 个面板，单击各自对应的按钮可以选择要查看的面板。使用图形面板，可以完成以下任务：

- 在图形窗口中添加二维或三维坐标系；
- 浏览和绘制工作空间变量的图形；
- 在图上添加标注。

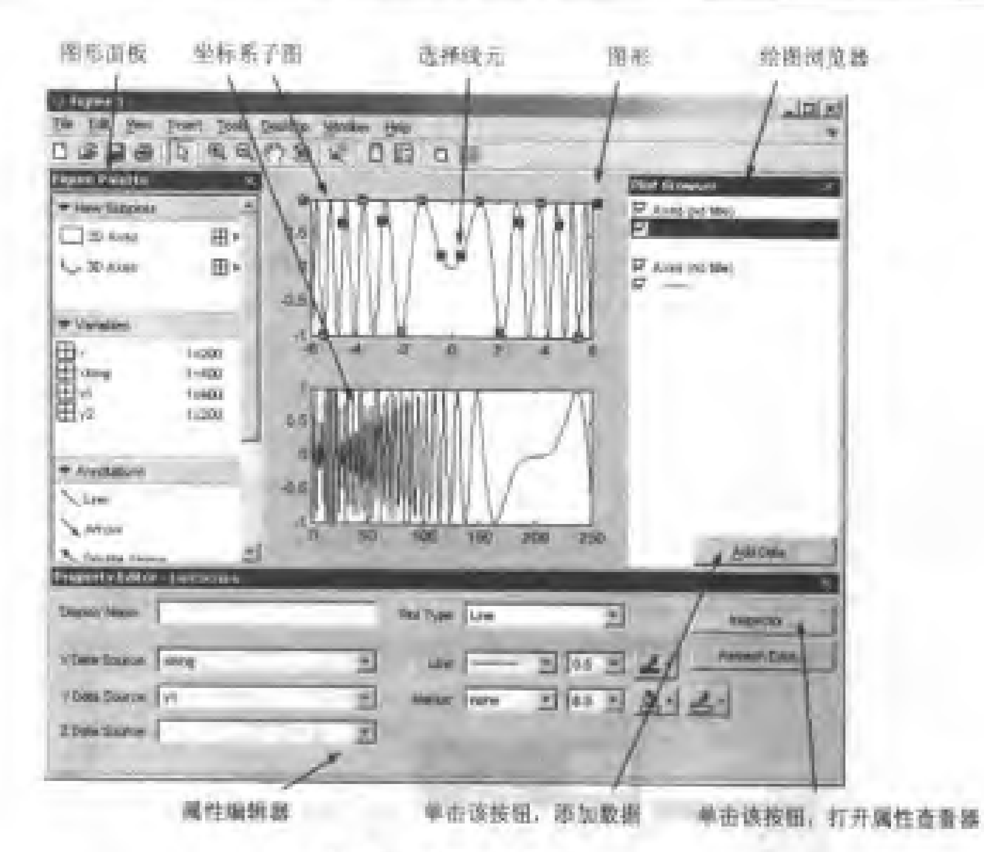



图 8-3　绘图工具界面

（1）添加子图坐标系

使用 “New Subplots” 面板，可以创建二维或三维坐标系网格。单击网格图标，显示选择器。MATLAB 显示选择器网格。

图 8-4 显示了 “New Subplots” 面板，设置为在图中显示 4 个相同大小的坐标系。已经存在的坐标系会自动改变大小，以便放下新输出的图形内容。

（2）绘制工作空间变量

“Variables” 面板显示当前的工作空间变量。在面板上双击一个变量，在数组编辑器中打开它。如果用鼠标右键单击变量，会弹出一个菜单，如图 8-5 所示，在其中选择绘图函数，绘制变量数据的图形。

例如，下面的图形演示如何绘制变量 z 的列数据，它等价于传递一个矩阵给 plot 函数。

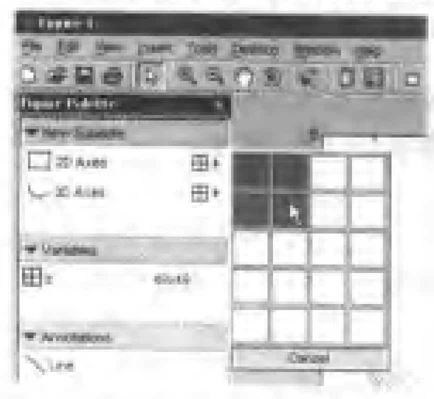


图 8-4　设置选择器网格



图 8-5　弹出式菜单

图 8-5 所示的上下文菜单包含各种可能的图形类型，它们基于选定的变量绘制，并可以对这些变量进行某些操作，例如在数组编辑器中进行打开、保存和复制等。

注意：选择不同的变量时上下文菜单中的选项可能会改变，因为某些特殊的变量可能会与有些图形类型不兼容。

可以直接将变量拖拉到坐标系中，此时 MATLAB 为该变量选择第一个合适的绘图类型。如果有多个坐标系，必须首先选定要绘图的坐标系，然后再把变量拖拉到该坐标系中。

在前面的例子中，用 plot 函数绘变量 z 数据的图形。如果从上下文菜单中无法获得必要的绘图函数，可以选择"More Plots"选项来显示"Plot Catalog"工具。

"Plot Catalog"工具提供获取大部分 MATLAB 图形函数的途径。可以在"Plotted Variables"文本框中键入任何工作空间变量，然后，它被作为变量传递给选定的绘图函数，用逗号间隔变量。也可以用图形面板中键入的任何工作空间变量输入 MATLAB 表达式。图 8-6 中显示了"Plot Catalog"工具，并描述它的组成元素。




图 8-6　"Plot Catalog"工具

（3）给图形添加标注

利用"Annotations"面板，可以在图中插入标注对象。添加对象，首先要选择准备添加的对象，然后通过用鼠标单击和拖拉来定位对象并改变对象的大小。

3．绘图浏览器

绘图浏览器提供了图中所有图形元素的图例，并列出了用于创建图形的所有坐标系和对象（直线、表面等）。例如，假设绘制 11×11 的矩阵 z 的图形，plot 函数会根据 z 中每一列的数据创建一条线元。

```
plot(z,'DisplayName','z')
```

设置 DisplayName 属性时，绘图浏览器会指出哪条直线与哪一列相对应。

如果要设置单条线元的属性，在绘图浏览器中双击该线元，它的属性显示在属性编辑器中，它打开并显示在图形窗口的底部。

可以在图中选择一个线元。选择后，该线元在绘图浏览器中的数据入口高亮显示，如图8-7所示。利用它，可以查看变量的哪一列数据生成了该线元。

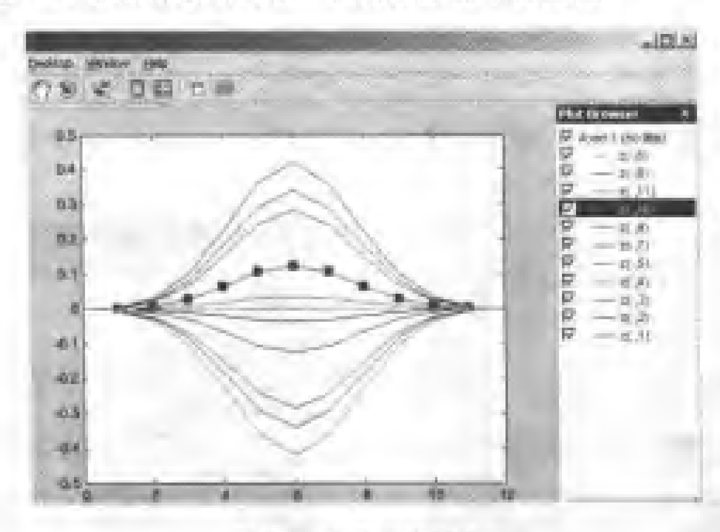


图8-7　在图中选择线元

绘图浏览器中每一项中的核选框控制对象的可见性。例如，假设只绘制z中某一列数据的图形，假设为正值，可以通过取消选择来取消选定列的图形显示。选定以后，图形会自动更新，如图8-8所示。

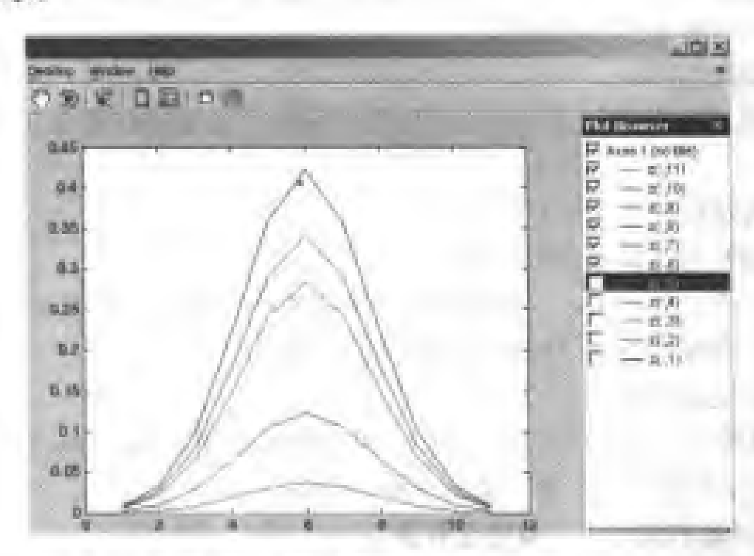


图8-8　进行设置以后自动更新图形

从右击上下文菜单中选择“Delete”选项，可以删除绘图浏览器中的所有选项。

绘图浏览器提供一种机制。通过该机制，可以给坐标系添加数据，操作过程如下：

（1）从“New Subplots”子面板中选择一个二维或三维坐标系；

（2）创建坐标系以后，在绘图浏览器中选定它，使得面板下面的“Add Data”按钮可用；

（3）单击“Add Data”按钮，显示“Add Data to Axes”对话框，如图8-9所示。

利用 “Add Data to Axes” 对话框，可以选择一种绘图类型，并指定工作空间变量传递给绘图函数。也可以指定一个 MATLAB 表达式，用它生成绘图数据。

选择工作空间变量创建图形：假设用三个工作空间变量来定义 ***Xdata,YData*** 和 ***ZData***，创建一个曲面图。

用 MATLAB 表达式创建图形：图 8-10 显示了 “Add Data to Axes” 对话框指定工作空间变量 *x* 的数据作为绘图的 *x* 轴数据，表达式 “x.^2+3*x+5” 的值作为 *y* 轴数据。



图 8-9 “Add Data to Axes” 对话框

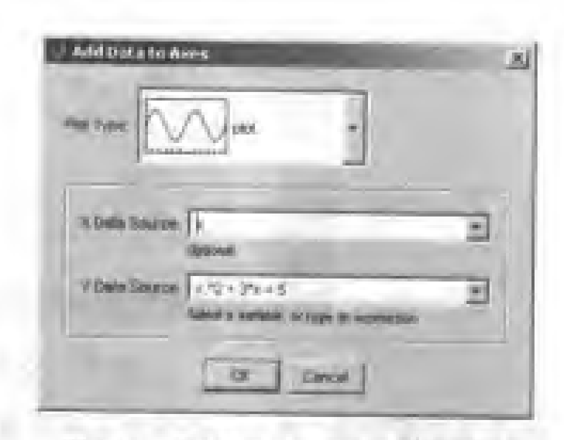


图 8-10 用 MATLAB 表达式创建图形

默认时将数据编号作为 *x* 数据。此时，MATLAB 根据 *x* 数据和 *y* 数据的编号绘图，它等价于调用只有一个变量的 plot 命令。

**4．属性编辑器**

利用属性编辑器可以获取选定对象的属性的子集，没有选择对象时，属性编辑器显示图形的属性。

打开属性编辑器有下面几种方法：

- 使用绘图编辑模式时，双击对象；
- 选择对象，用右键单击它，然后在上下文菜单中选择 “Properties” 选项；
- 从 “View” 菜单中选择 “Property Editor” 选项；
- 使用 propertyeditor 命令。

可以用属性编辑器改变绘图类型，例如，可以通过改变 “Plot Type” 中的选项将图 8-11 中的线形图改为火柴杆图、阶梯图、面积图和条形图。

**5．获取所有对象的属性——属性查看器**

使用属性编辑器可以改变大部分常用的对象属性。如果想获取所有对象属性，使用属性查看器。在任何属性编辑器面板上单击 “Inspector” 按钮，可以显示属性查看器。图 8-12 为属性查看器，它显示图 8-11 中线元的属性。

如果想获取光照或控件上下文菜单对象的属性，需要用 MATLAB 命令获取对象的句柄，因为无法单击这些对象。例如，下面用 findobj 命令获取当前坐标系中所有光照对象的句柄。

```
h = findobj(gca,'Type','light');
```

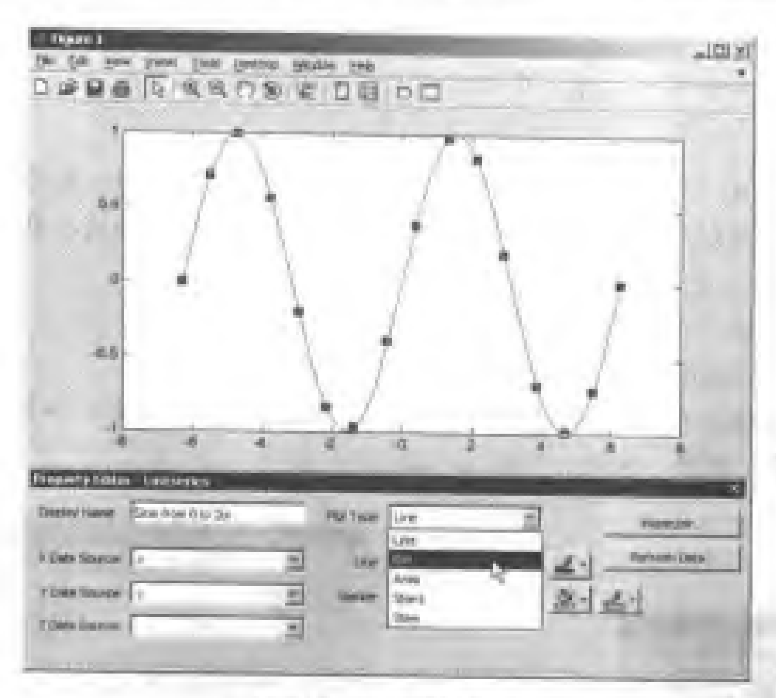


图 8-11 改变图形类型

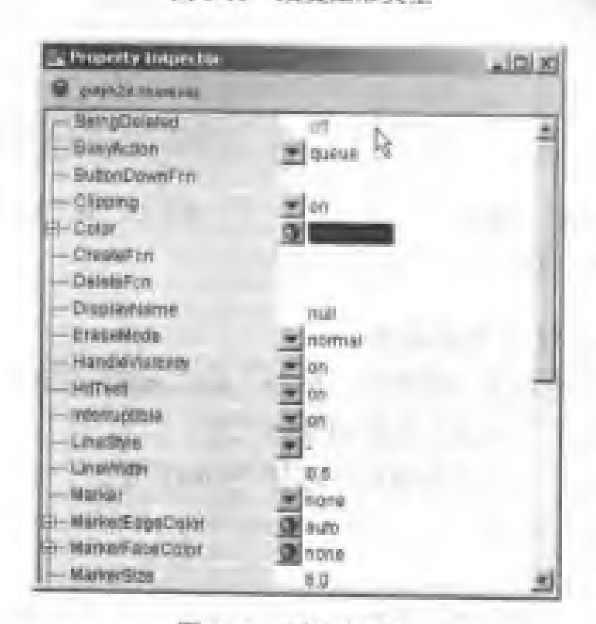



图 8-12 属性查看器

然后用 inspect 命令显示属性查看器。

```
inspect(h) % 查找所有光照对象
inspect(h(1)) % 查找列表中的第一个光照对象
```

### 8.1.3 使用绘图工具

下面的例子演示如何利用绘图工具绘制工作空间变量与输入 "Add Data to Axes" 对话框中的表达式的图形。

- 在工作空间中创建一个变量：

```
x = -2*pi:pi/25:2*pi;
```

- 然后用 plottools 命令创建一个有绘图工具的图形窗口：

```
plottools
```

- 在图形面板的“New Subplot”面板中单击“2D Axes”选项，如图 8-13 所示。

坐标系显示出来以后，绘图浏览器中的“Add Data”按钮被激活，单击这个按钮，显示“Add Data to Axes”对话框，作如下设置：

- 在“Plot Type”下拉式列表框中选择“plot”；
- 在“X Data Source”下拉式列表框中选择“X”；
- 在“Y Data Source”下拉式列表框中选择“sin(x).^2”，如图 8-14 所示；
- 单击“OK”按钮，绘图。




图 8-13　在“New Subplot”面板中单击“2D Axes”选项

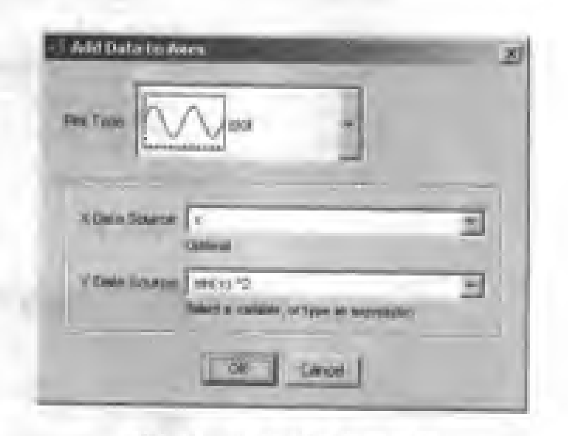



图 8-14　进行绘图数据设置

MATLAB 绘制 sin(x).^2 与 X 的图形。

现在在同一坐标系中添加另一幅图形，单击“Add Data”按钮，指定绘图数据：

- 在“X Data Source”下拉式列表框中选择“X”；
- 在“Y Data Source”下拉式列表框中选择“sin(x).^8”；
- 单击“OK”按钮，绘图。

绘图结果如图 8-15 所示。

用“New Subplots”面板在当前坐标系下方添加第二个坐标系。单击“2D Axes”后面的向右箭头键，移动鼠标，使两个方块变黑，竖向排列，如图 8-16 所示。该操作将在已经存在的坐标系下方创建一个子图坐标系。MATLAB 改变已经存在的坐标系的大小，使得两个坐标系都正好在图形窗口中放下。

MATLAB 插入新的坐标系以后，在绘图浏览器中选择它的入口，然后单击“Add Data”按钮，显示“Add Data to Axes”对话框以后，作如下设置：

- 在“X Data Source”下拉式列表框中选择“X”；
- 在“Y Data Source”下拉式列表框中选择“sin(x).^3”；
- 单击“OK”按钮，绘图。

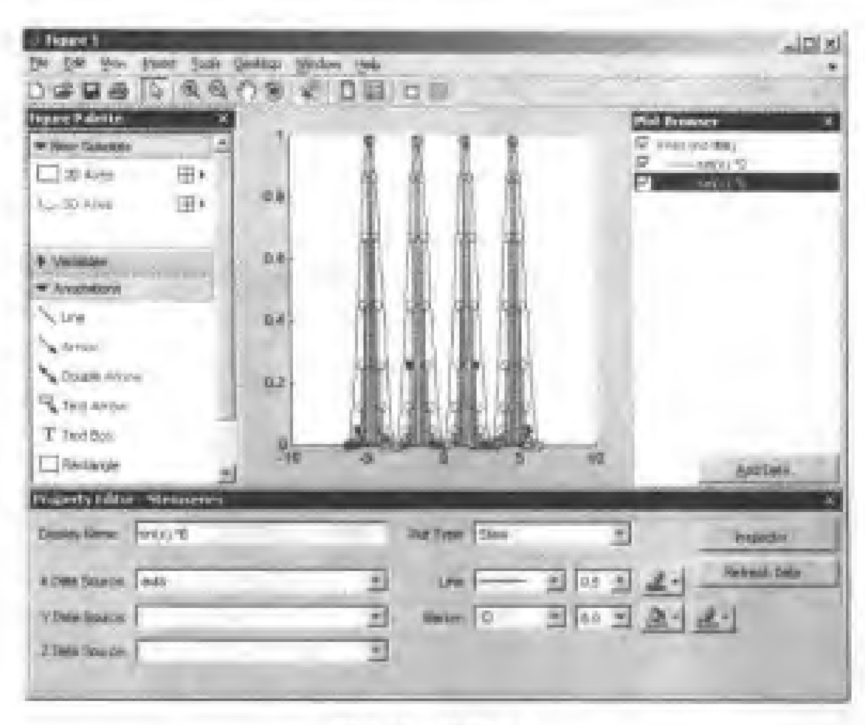


图 8-15 绘图结果

单击“Add Data”按钮并指定数据，添加另一幅图形：

- 在“X Data Source”下拉式列表框中选择“X”；
- 在“Y Data Source”下拉式列表框中选择“sin(x).^9”；
- 单击“OK”按钮，绘图。

在绘图浏览器中选择第二个坐标系下面标注为“sin(x).^9”的图形，将属性编辑器中的“Plot Type”设置为“Area”。



图 8-16 添加子图坐标系

用属性编辑器调整两个坐标系中的 X 轴：

- 在绘图浏览器中选择第一个坐标系；
- 将“X Limits”改为-7 和 7。

对第二个坐标系重复此步骤。

在绘图浏览器中选择第一个坐标系，并在属性编辑器中设置下面的属性：

- 将“Title”属性值设置为“Even Powers”；
- 将“X Label”设置为“X”；
- 单击“Y Axis”选项卡并将“Y Label”设置为“Sine of X”。

在绘图浏览器中选择第二个坐标系，并在属性面板中设置下面的属性：

- 将“Title”属性值设置为“Odd Powers”；
- 将“Axis label”设置为“X”；
- 在“Y Axis”选项卡上将“Axis label”设置为“Sine of X”。

注意，绘图浏览器中显示了新的坐标系名称。图 8-17 显示以上操作的最终结果。

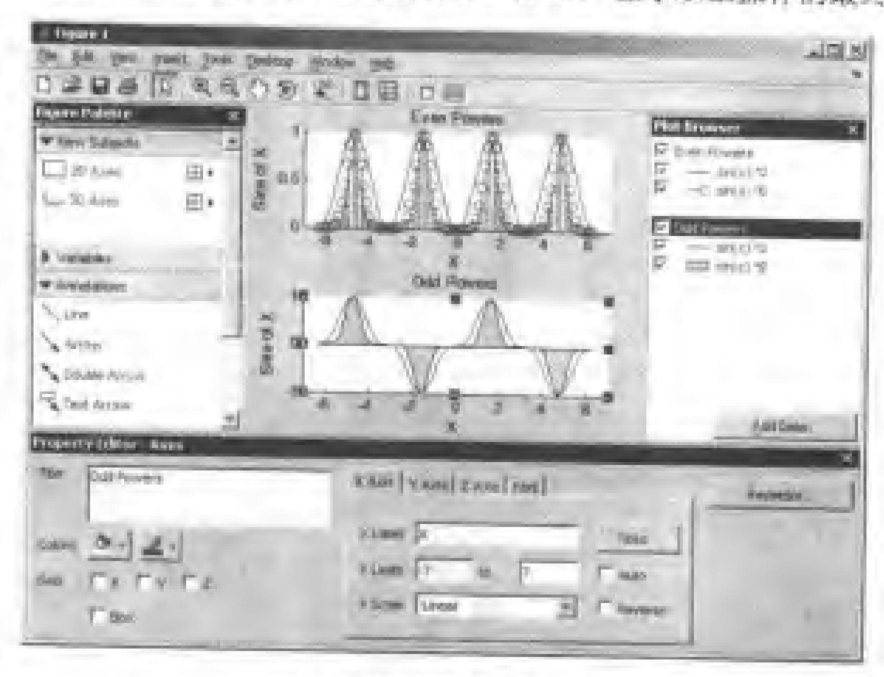


图 8-17　最终绘图结果

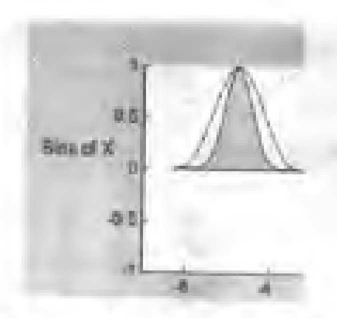


图 8-18　改变位置以后的文本标签

在第一个坐标系中选择 Y 轴标签的文本（现在在绘图浏览器中标注为“Even Powers”），并单击属性编辑器中的“Inspector”按钮，将“Rotation”属性设置为 0，并手工重新设置文本的位置。为了使 Y 轴标签间隔稍宽，可以选择坐标系并用鼠标把它向右移动。

对第二个坐标系重复此过程。

改变位置后的文本标签如图 8-18 所示。

### 8.1.4　用工作空间中的变量绘图

下面演示如何用图形面板选择变量进行绘图。假设用下面的语句创建 3 个变量，它们位于工作空间中，名为 x, y 和 z。

```
[x,y] = meshgrid([-2:.2:2]);
z = x.*exp(-x.^2-y.^2);
```

按照下面的步骤将该数据显示为曲面图或等值线图。

（1）用 figurepalette 命令打开带图形面板工具的图形窗口：

```
figurepalette
```

扩展“Variables”面板，并在按住“Shift”键的同时单击准备传给绘图函数的第三个变量。因为 Surfc 命令不在列表中，选择“More Plots”按钮，如图 8-19 中所示。

在“Plot Catalog”工具中，在“Categoreis”列表框中选择“3D Surfaces”选项；在“Plot Types”列表框中选择“Surfc”选项，如图 8-20 所示。单击“Plot”按钮，创建图形。

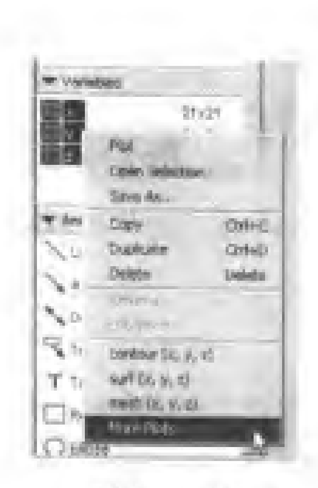


图 8-19　菜单

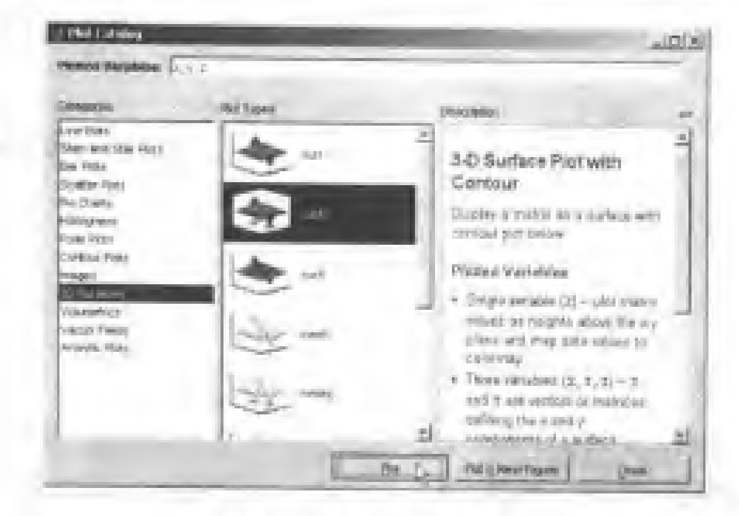


图 8-20　"Plot Catalog" 工具

生成的图形如图 8-21 所示。

可以在 "Plot Catalog" 工具中输入 MATLAB 表达式或变量，假设已经在工作空间中创建了下面的变量。

```
t = 0:.01:20;
alpha = .055;
```

而且希望用 t 和下面的表达式绘图。

```
exp(-alpha*t).*sin(.5*t)
```

用 figurepalette 命令打开带图形面板工具的图形窗口。

```
figurepalette
```

首先选择变量 *t*，用鼠标右键单击它，显示上下文菜单，选择 "More Plots..." 选项，如图 8-22 所示。

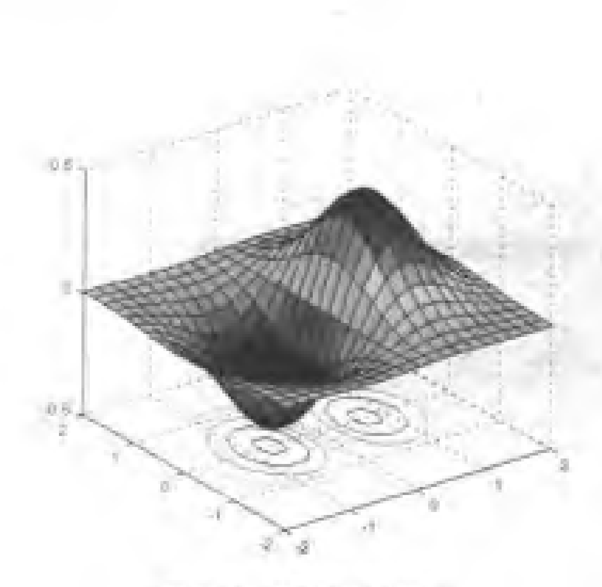


图 8-21　最后生成的图形



图 8-22　在菜单中进行选择

“Plot Catalog”工具显示出来以后，在“Plotted Variables”文本框中添加表达式，如图 8-23 所示。注意，可以引用变量 alpha，因为在基本工作空间中创建了它。

单击“Plot”按钮创建图形。注意，上一步操作的效果与下面命令行的等价。

```
plot(t,exp(-alpha*t).*sin(.5*t))
```

图 8-23　在“Plot Catalog”工具中添加表达式

### 8.1.5　指定数据源

利用图形对象的属性可以指定定义对象的数据源。例如，可以把工作空间中的变量名或 MATLAB 表达式（见图 8-24）作为直线对象的 XData Source,YData Source 或 ZData Source 属性的值来指定。然后可以用属性编辑器改变变量名或更换表达式，图形随即更新。

首先，在命令窗口键入下面的命令行，定义两个变量。

```
t = 0:.01:20;
alpha =.055;
```

然后，用 $t$ 和 exp(-alpha*t).*sin(.5*t)数据绘图，用 plot 函数或绘图工具进行绘制。

创建图形以后，可以用属性编辑器改变数据源：

- 双击线元，显示它的属性面板；
- 在“Y Data Source”文本框中输入 MATLAB 表达式。

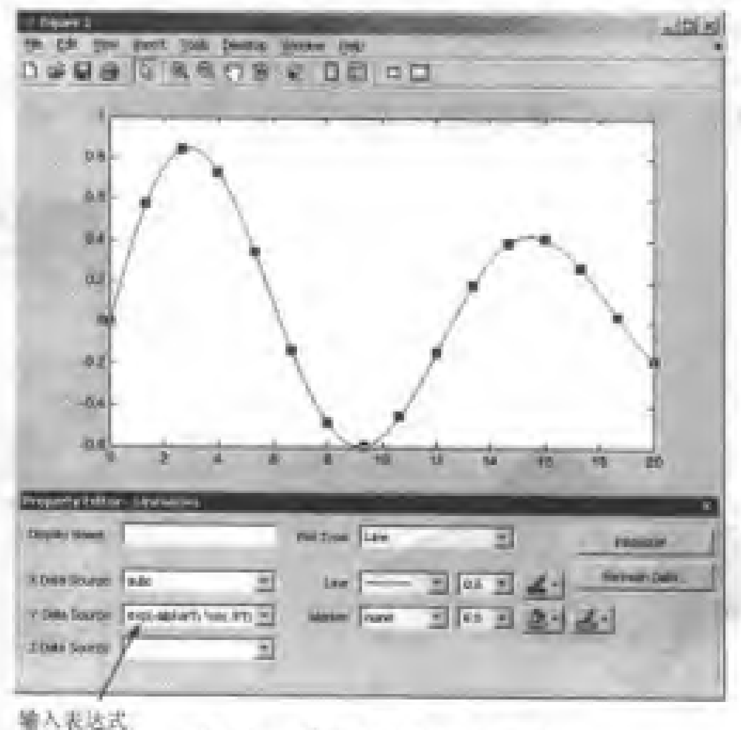



图 8-24　输入 MATLAB 表达式

现在可以在“Y Data Source”文本框中修改表达式，并查看图形如何改变了。改变文本以后，单击“Refresh Data”按钮，更新数据。

在图 8-25 中，alpha 不再取消，所以函数图形的形状呈现增长趋势，将 sin(.5*t)改变为 sin(1.5*t)以后，周期也缩短了。

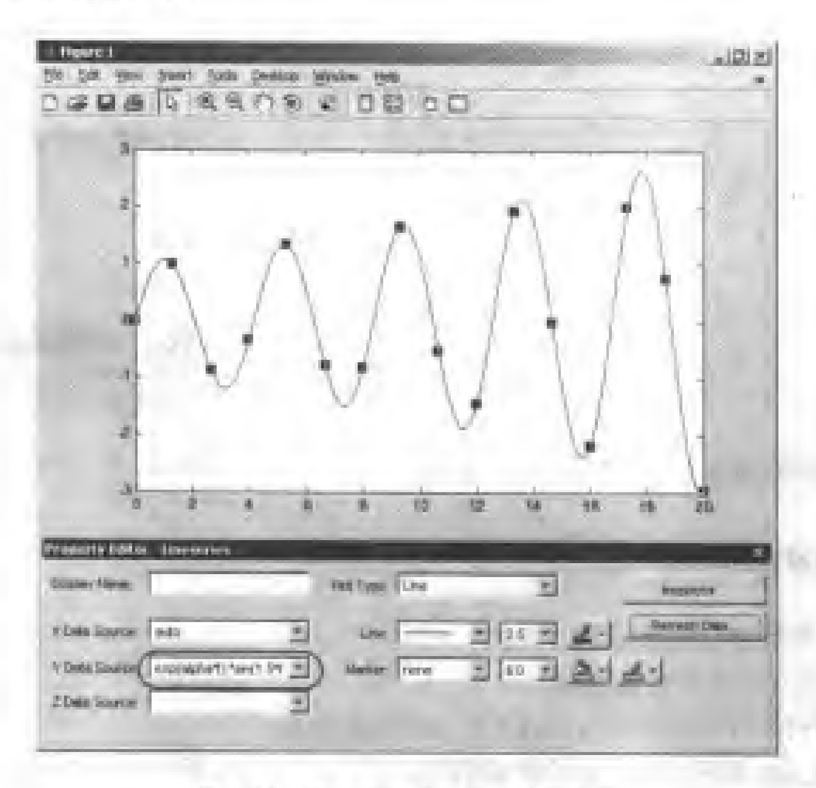


图 8-25　修改表达式以后的绘图效果

假设用矩阵数据创建线形图，例如，

```
z = peaks;
h = plot(z,'YDataSource','z');
```

因为 MATLAB 为 z 中的每个列创建一个线形对象，所以，h(1)的数据源是 z(:,1)，h(2)的数据源是 z(:,2),…, h(n)的数据源是 z(:,n)。

下面的例子通过创建 M 文件重建图形。假设用下面的代码创建图形：

```
t = 0:.2:20;
alpha =.055;
stem(t,exp(-alpha*t).*sin(5*t))
```

然后用属性编辑器修改生成的图形，使之如图 8-26 所示。

从"Figure"菜单中选择"Generate M-File"选项（如图 8-27 所示），可以生成代码来重建图 8-26。MATLAB 创建一个函数来重建图形并在 M 文件编辑器中打开生成的 M 文件。这个特点在捕获属性设置和用绘图工具所作的其他修改时很有用。

必须给函数提供数据变量 *t* 和 exp(-alpha*t).*sin(5*t)。生成的函数不会保存必要的数据来重建图形。

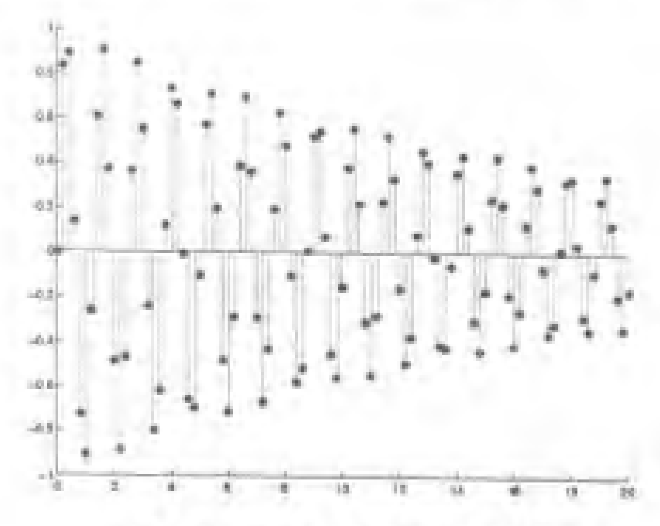


图 8-26　用属性编辑器修改生成的图形

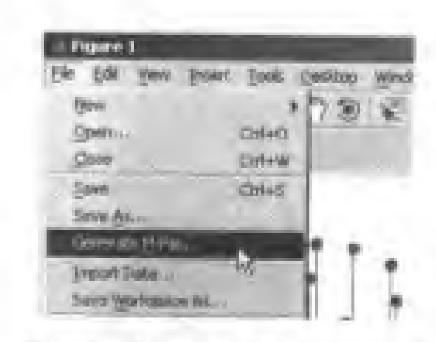



图 8-27　在“File”菜单中进行选择

### 8.1.6　编辑图形

MATLAB 支持两种图形编辑方式：

- 用鼠标交互选择和编辑对象；
- 在命令行或 M 文件中用 MATLAB 函数编辑对象。

如果 MATLAB 图形窗口中的绘图编辑模式可用，可以对图形进行点击编辑。使用这种模式，可以通过双击对象并改变它的属性值来修改图形的外观，可以用属性编辑器获取属性。

如果更喜欢从命令行进行操作，或者正在创建 M 文件，可以用 MATLAB 命令编辑所创建的图形。利用 MATLAB 句柄图形系统，可以用 set 和 get 命令改变图中对象的属性。

### 8.1.7　使用图形编辑模式

单击工具条中的箭头键，可以启用绘图编辑模式。在绘图编辑模式下，可以对图形进行交互编辑，包括添加图例、文本、箭头，选择图形元素，定位图例、文本等，如图 8-28 所示。

注意，绘图编辑模式提供了获取 MATLAB 图形对象属性的替代方式。但是，只能通过这种机制获取对象属性的子集。有时候可能需要组合使用交互编辑和命令行编辑两种形式来获得想要的结果。

MATLAB 图形窗口支持点击编辑模式，可以用该模式定制图形的外观。下面介绍如何启用绘图编辑模式并完成基本的编辑任务，包括：

- 选择图形中的对象；
- 剪切、复制和粘贴对象；
- 移动对象和改变对象的大小；
- 设置对象属性；
- 保存工作。

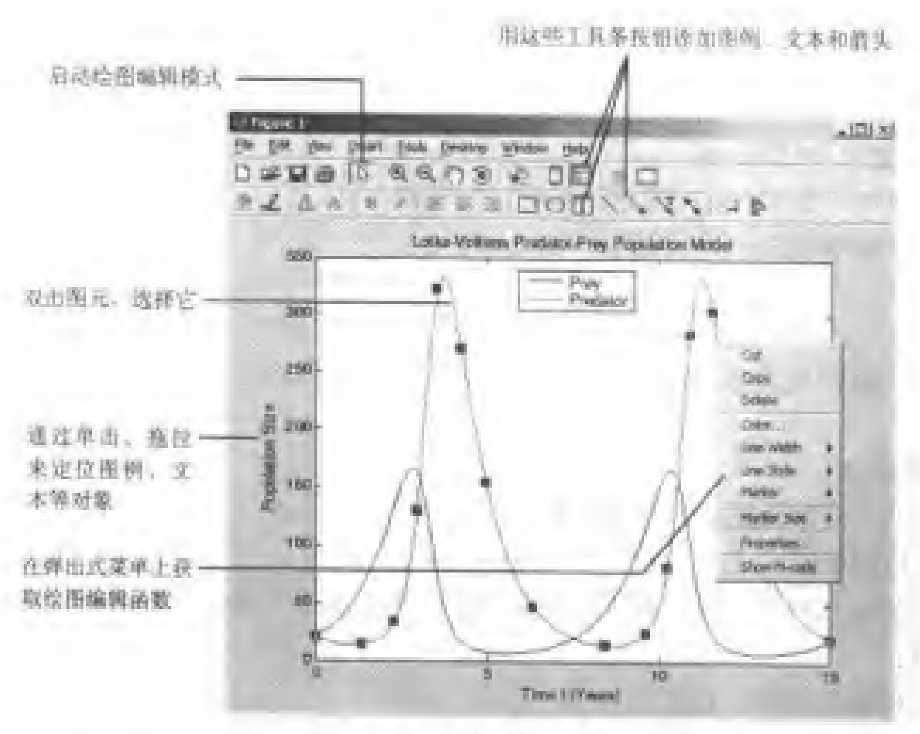



图 8-28　图形编辑工具和手段

### 1. 启用绘图编辑模式

用单击方式选择图形以前，需要首先激活绘图编辑模式。有几种方法可以激活绘图编辑模式：

- 在“Tools”菜单中选择“Edit Plot”选项；
- 在图形窗口的工具条中单击选择按钮，启用绘图编辑模式；
- 从“Edit”或“Insert”菜单中选择一个选项，例如，在“Edit”菜单中选择“Axes Properties”选项，则 MATLAB 激活绘图编辑模式，坐标系以选中模式显示；
- 在 MATLAB 命令窗口中运行 plotedit 命令；
- 用 plottools 命令启动绘图工具。

图形窗口处于绘图编辑模式时，“Tools”菜单中的“Edit Plot”选项被选中，并且工具条中的选择按钮被按下。

### 2. 退出绘图编辑模式

单击选择按钮或“Tools”菜单中的“Edit Plot”选项，退出绘图编辑模式。取消该模式时，选择按钮弹起。

### 3. 选择多个对象

按照下面的步骤选择多个对象：

（1）启用绘图编辑模式；

（2）把鼠标移到对象上，按住 Shift 键的同时选择它。重复此操作，选择所有要选择的对象。

可以对所有选中的对象进行进一步的编辑。例如，要从图中删除文本框标注和箭头标注，可以选择对象，然后从“Edit”菜单中选择“Cut”选项。

**4. 取消对象的选择**

取消对象的选择，把鼠标光标移向图形窗口背景并单击左键，也可以用 Shift 单击方式取消对象的选择。

**5. 剪切、复制和粘贴对象**

从图形中剪切、复制和粘贴对象，按照以下步骤进行：

- 启用绘图编辑模式；
- 选择对象；
- 从“Edit”菜单中选择“Cut”、“Copy”或“Paste”选项，或者使用标准快捷键。

同样，绘图编辑模式可用时，可以右击对象，然后从弹出的上下文菜单中选择一个编辑命令。

**6. 移动和缩放**

按照下面的步骤对对象进行移动和缩放：

（1）启用绘图编辑模式；

（2）只对坐标系对象有效：右击空白区域并从弹出的上下文菜单中选择“Unlock Axes Position”选项；

（3）选择对象上出现的手柄。

拖拉对象，可以把对象移到新位置。拖拉对象上的手柄，可以改变对象的大小。

**7. 设置对象的属性**

在 MATLAB 中，图形中的每个对象都支持一系列的属性，它们控制图形的外观和行为。例如，线形对象具有控制粗细、颜色和线型的属性；双击对象，显示属性编辑器。要编辑坐标系或图形窗口的属性，双击不包含对象的空白区域。

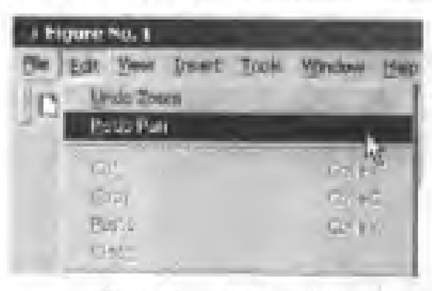



图 8-29 “Edit”菜单

**8. 取消和重复操作**

利用“Edit”菜单中的“Undo”和“Redo”选项可以取消和重复操作。例如，创建一幅图形，放大它，平移视图，然后取消平移操作，菜单如图 8-29 所示。

现在可以取消前面的放大操作或重复刚刚取消的平移操作。

## 8.1.8 保存结果

编辑图形以后，可以把结果保存为其他 MATLAB 进程和其他应用程序可以使用的格式。

**1. 以 MAT 格式保存图形**

MATLAB 支持二进制格式，可以将图形保存为该格式，并在后面的 MATLAB 进程中打开。MATLAB 指定这些文件的扩展名为.fig。

按照下面的步骤保存图形：

（1）从“File”菜单中选择“Save”选项，或者在工具条中单击“Save”按钮。如果是第一次保存，则打开“Save As”对话框；

（2）确保“Save as type”选项设置为“Fig-File”;

（3）指定文件路径;

（4）单击“OK”按钮。

图形保存为.fig 文件，它是二进制的。

**2. 打开图形文件**

按照下面的步骤打开图形文件：

（1）从“File”菜单中选择“Open”选项，在工具条上单击“Open”按钮;

（2）选择要打开的图形文件，并单击“OK”按钮。

图形文件显示在新的图形窗口中。

**3. 保存为不同的格式**

将图形保存为其他应用程序可用的格式，如 TIFF 或 EPS 等，按照下面的步骤进行：

（1）从“File”菜单中选择“Export Setup”选项，打开的对话框中提供指定大小、字体、线宽和线型等格式的选项;

（2）在“Export Setup”对话框中选择“Export”按钮，显示标准的“Save As”对话框;

（3）从“Save as type”下拉式列表框中选择要保存的格式;

（4）输入文件名;

（5）单击“Save”按钮。

**4. 将图形复制到剪贴板**

（1）在“Edit”菜单中选择“Copy Options”选项，显示“Preferences”对话框中的“Copying Options”选项卡;

（2）填完“Copying Options”选项卡中的文本框，单击“OK”按钮;

（3）从“Edit”菜单中选择“Copy Figure”选项。

**5. 生成 M 文件来重建图形**

可以根据图形创建 M 文件，然后用该文件重建图形。这种方法适合于创建用绘图工具创建的图形的 M 代码。按照下面的步骤进行操作：

（1）从“File”菜单中选择“Generate M-file”选项;

（2）MATLAB 在编辑器中显示生成的代码;

（3）用“File”菜单中的“Save As”选项保存 M 文件。

**6. 运行保存后的 M 文件**

大部分生成的 M 文件需要将数据作为变量输入。M 文件假设使用与原来相同的数据进行绘图。M 文件开始的说明，介绍所希望输入的数据的类型。例如，下面的语句演示了需要 3 个输入向量的情况。

```
function createplot(X1, Y1, Y2)
%CREATEPLOT(X1,Y1,Y2)
%  X1:  vector of x data
%  Y1:  vector of y data
%  Y2:  vector of y data
```

## 8.2 数据查看工具

确定表示数据的最佳图形类型以后，可以用本节介绍的工具进一步增强信息的可视显示效果。使用这些工具，可以交互查看数据，不再需要通过用 MATLAB 命令设置图形属性来达到相同的效果。

一旦获得所需要的结果，就可以生成 MATLAB 代码，它对于重建前面交互创建的图形来说是必要的。

数据查看工具包括数据光标、缩放、平移和旋转等几种。

### 8.2.1 数据光标——交互显示数据的值

使用数据光标时，能直接显示光标处直线、曲面和图像上点的值。数据光标模式可用时，可以实现下面的显示：

- 单击数据值定义的任何图形对象，可以显示最近点处的 $x$, $y$ 和 $z$ 值；
- 数据点之间的插值数据；
- 在图上显示多个数据提示；
- 在光标窗口中显示数据值；
- 作为工作空间变量输出数据值；
- 打印或输出图形数据提示或光标窗口作为标注显示。

**1．启用数据光标模式**

在图形工具条上选择数据光标图标或者在“Tools”菜单中选择“Data Cursor”选项，启用数据光标模式。

一旦数据光标模式可用，单击线元或其他图形对象，返回单击点的数据值。例如，图 8-30 中，黑色的方框表示选中点，文本标注表示该处的数据值。

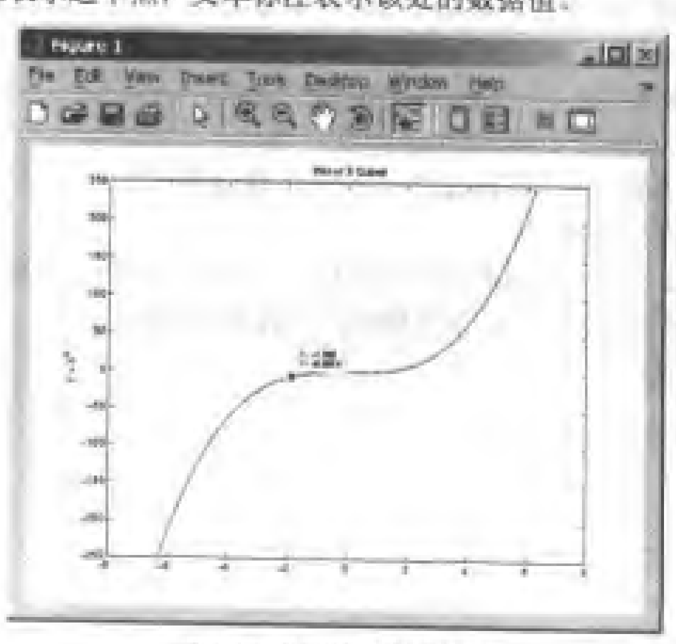



图 8-30　线形图上的选中点

2．移动文本标注

可以用箭头键和鼠标键移动文本标注，向上和向右箭头键把标注向数据数组中索引值更大的数据点移动，向下和向右箭头键则相反。

3．数据提示

可以将数据提示文本框放在点的右上部、左上部、左下部或右下部。定位时，拖拉文本框就行了。也可以用鼠标移动数据提示文本框。图 8-31 和图 8-32 分别示出拖拉标注文本和图像上点的标注。

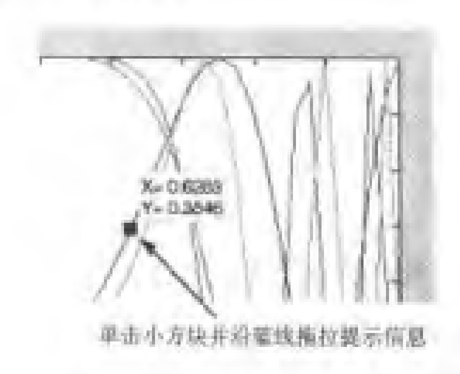



图 8-31　拖拉标注文本

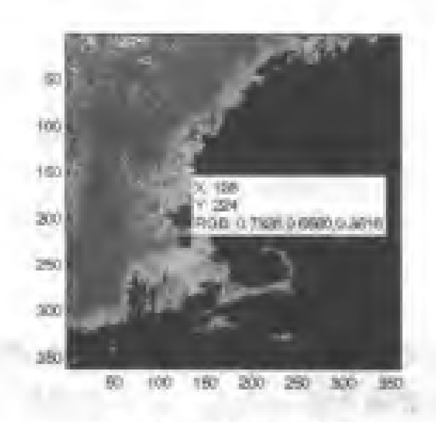


图 8-32　图像上点的标注

单击数据提示文本框并拖拉，可以将它移动到图中不同的位置，也可以用箭头键移动数据提示文本框。

图像上的数据提示显示点的 $x, y$ 坐标和 RGB 值。

可以用数据提示读取三维图形上的数据点。在三维视图中，数据提示显示 $x, y$ 和 $z$ 坐标，如图 8-33 所示。

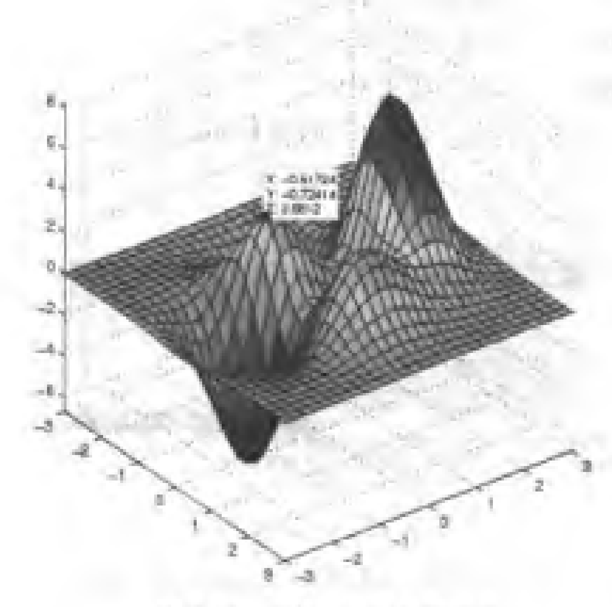


图 8-33　三维图上的数据提示

正常情况下，一次只显示一个数据提示。但是，也可以同时显示多个数据提示。这为在图中标注多个点提供了一种简便途径。

按下面的步骤显示多个数据提示：

（1）使数据光标模式可用，光标变为十字丝形；

（2）单击图形，插入一个数据提示；

（3）右键单击图形，显示上下文菜单；选择“Create New Datatip”选项，如图 8-34 中所示。

（4）单击图形，放置第二个数据提示，如图 8-35 中所示。

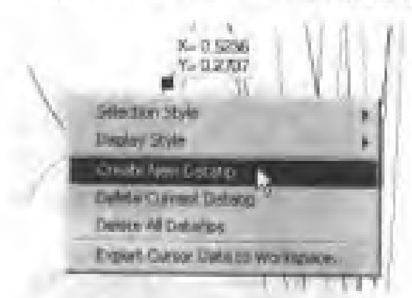



图 8-34　在菜单中选择选项

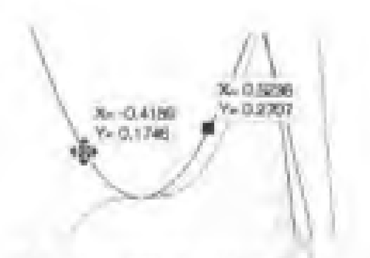



图 8-35　放置第二个数据提示

可以用 datacursormode 函数定制数据光标显示的文本。

可以删除大部分最近添加的数据提示或所有数据提示。处于数据光标模式时，右击图形显示上下文菜单。

- 选择“Delete Current Datatip”选项，删除最后添加的数据提示；
- 选择“Delete All Datatips”选项，删除所有数据提示。

4. 显示类型——数据提示或光标窗口

默认时，数据光标把值作为数据光标显示，也可以把数据值显示在光标窗口中。可以在图中放置多个数据提示，这有助于图形标注。

要使用光标窗口，需要将显示类型作如下改变：

（1）在数据光标模式下，右击图形显示上下文菜单；

（2）将鼠标移到“Display Style”选项上；

（3）选择“Window Inside Figure”选项，如图 8-36 所示。

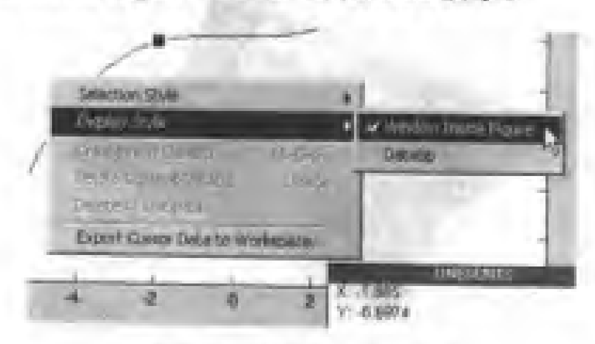



图 8-36　在菜单上进行选择

默认时，数据光标显示距离鼠标单击点最近的数据点的值，并且数据标记跟着跳到这个点上。根据距离鼠标单击点最近的两个数据点进行线性插值，数据光标还可以确定图形定义点之间某个点的值。

如果希望选择沿图形分布的任何点并显示它的值，按下面的步骤进行：

（1）处于数据光标模式时，右击图形，显示上下文菜单；

（2）把鼠标移至“Selection Style”选项；

（3）选择“Mouse Position”选项，如图 8-37 所示。




图 8-37　在菜单上选择选项

注意，使用箭头键时插值模式无效。

**5．将数据值输出到工作空间变量**

可以把数据光标显示的值输出到 MATLAB 工作空间变量中。要实现此操作，在数据光标模式下显示右击上下文菜单，并选择“Export...”选项，如图 8-38 所示。

然后，MATLAB 显示如图 8-39 所示的对话框，可以用它命名工作空间变量。

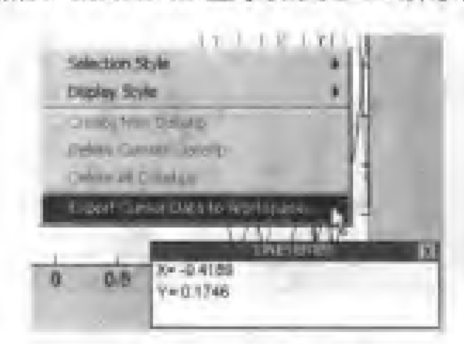



图 8-38　选择选项




图 8-39　“Output Data Cursor Information”对话框

单击“OK”按钮时，MATLAB 用基本工作空间中的指定名称创建一个结构。该结构包含下面的字段：

- Target　包含数据点的图形对象的句柄；
- Position　数据光标位置上的 $x, y$ 坐标值。

线元和线元序列对象还多一个字段：

- DataIndex　是数据数组中的标量索引号，对应于最近的数据点，每个数据中的值相同。

例如，保存的工作空间变量为 cursor_info，则可以通过引用 Position 字段获得位置数据，即

```
cursor_info.Position
ans =
      0.4189 0.1746 0
```

## 8.2.2　二维和三维图形的缩放

缩放可以改变图形的视图效果。放大图形以后，可以看到更小区域内的细节内容。二维或三维视图有不同的缩放方式。

单击按钮可以启用缩放操作。

### 1. 放大二维视图

在二维视图中，单击坐标系中要放大的区域，或者拖拉出一个包围局部面积的矩形，可以进行放大操作。MATLAB 会重绘坐标系，同时改变范围来显示指定的区域。

例如，在图 8-40 中选择要放大的区域：

生成一个改变大小后的坐标系，其中只显示选定的区域，如图 8-41 所示。

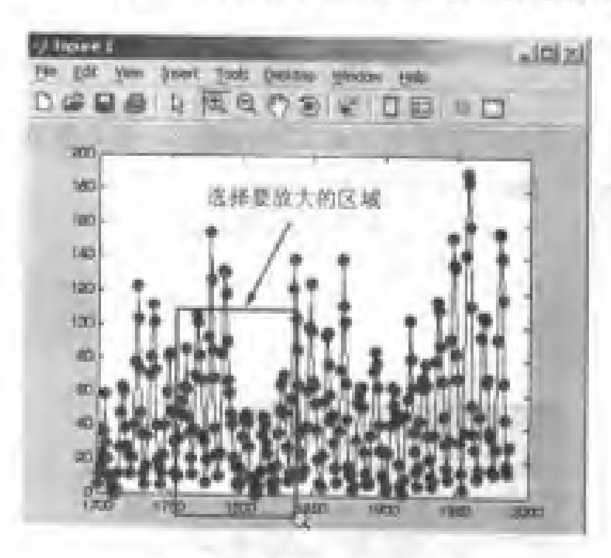



图 8-40　选择要放大的区域

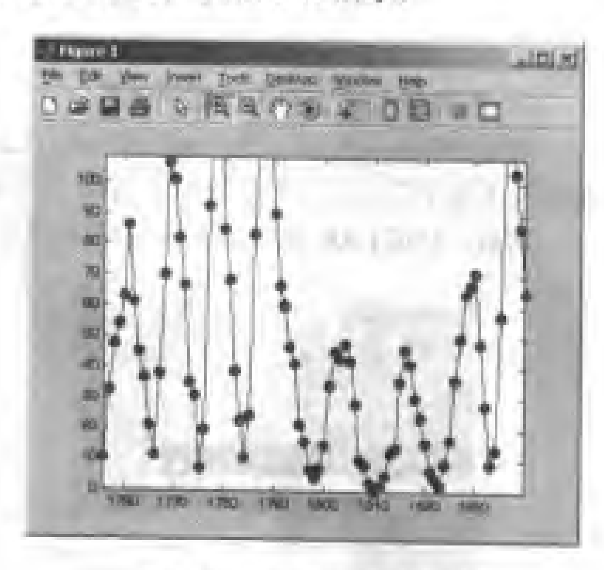


图 8-41　放大显示选定的区域

### 2. 水平缩放或竖向缩放

在二维视图中，可以将缩放固定在水平方向或垂直方向上。具体操作是在缩放模式下右击图形显示上下文菜单，然后选择必要的缩放选项，如图 8-42 所示。




图 8-42　在菜单上选择缩放选项

### 3. 三维视图的缩放操作

三维视图中，向上或向右移动光标放大图形，向下或向左移动则缩小图形。三维缩放不会改变坐标系的范围，而是改变视图，就像用变焦相机看到的效果一样。图 8-43 是对图 8-33 中的局部进行放大以后的显示效果。

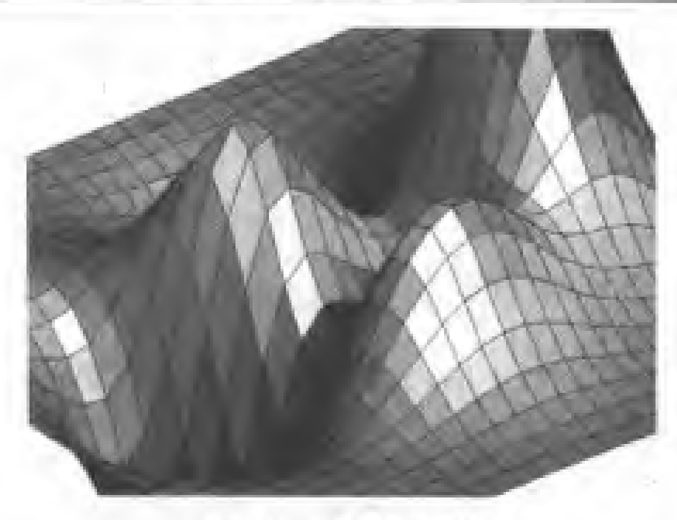


图 8-43　三维图形的放大

### 8.2.3　平移图形

可以用平移工具向上、向下或向右、向左移动图形视图。

二维平移可以改变坐标系范围的显示效果，但不改变图形实际的坐标范围。例如，假设有一个时间序列波形，现在希望放大它的同时可以看到整幅图，可以按照图 8-44 中的步骤进行操作。

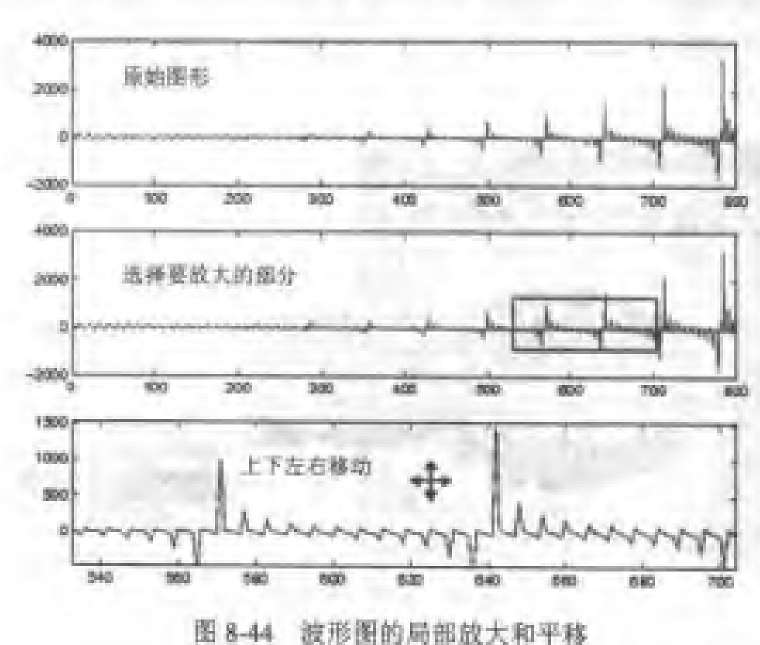



图 8-44　波形图的局部放大和平移

### 8.2.4　三维视图的交互旋转

MATLAB 中可以将图形旋转到任何方位。旋转牵涉到坐标系和坐标系中所有图形对象的重新定位，所以旋转时对图形对象的数据没有影响，只是 $x$, $y$ 和 $z$ 坐标相对于观察者的方位改变了。

### 1. 启用旋转三维模式

有 3 种方法可以启用旋转三维模式：

- 从"Tools"菜单中选择"Rotate 3D"选项；
- 在图形工具条中单击按钮；
- 运行 rotate3d 命令。

启用模式以后，在图形上拖拉鼠标就可以实现旋转。

### 2. 选择预定义的视图

旋转三维模式启用时，可以从右击上下文菜单中控制不同的旋转选项，可以在右击菜单中旋转至预定义的视图。

可以从右击上下文菜单的"Rotation Options"子菜单中选择两种旋转类型中的一种。

- Plot Box Rotate　只显示坐标系的包围盒，以加速复杂对象的旋转。如果默认的"Continuous Rotate"类型不能接受，使用本选项。
- Continuous Rotate　旋转时显示所有图形。

### 3. 旋转时的坐标系行为

旋转时可以选择两种与坐标系纵横比有关的行为类型。

- Stretch-to-Fill Axes　默认的坐标系行为，适合于二维图形。选择这种方式，旋转时会自动改变坐标系的纵横比，使图形填满图形窗口范围。
- Fixed Aspect Ratio Axes　保持对象的形状不变，适合于三维图形的显示。

图 8-45 显示了选择"Stretch-to-Fill Axes"选项时一个圆球的旋转效果。注意，图中圆球看起来像个椭球体。

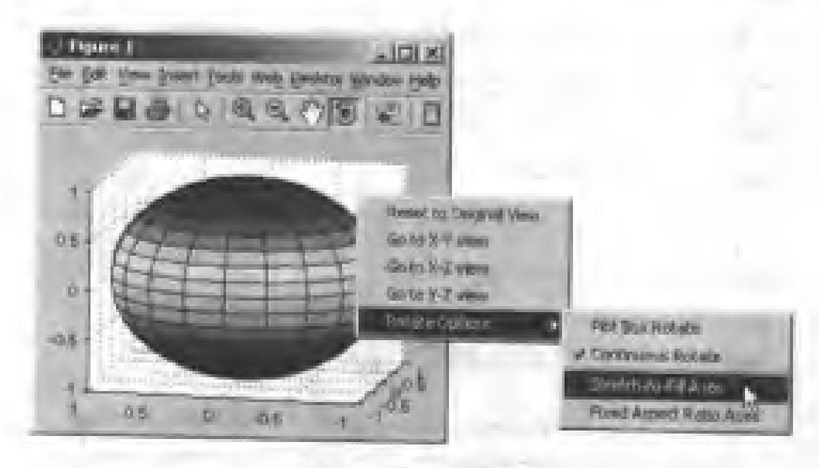



图 8-45　选择旋转选项

图 8-46 为选择"Fixed Aspect Ratio Axes"选项时球体的显示效果，现在旋转时球体会保持形状不变。

## 8.2.5　分析图形数据

可以用下面的工具直接完成某些分析任务：

- 基本曲线拟合；
- 在图形上添加基本统计图。

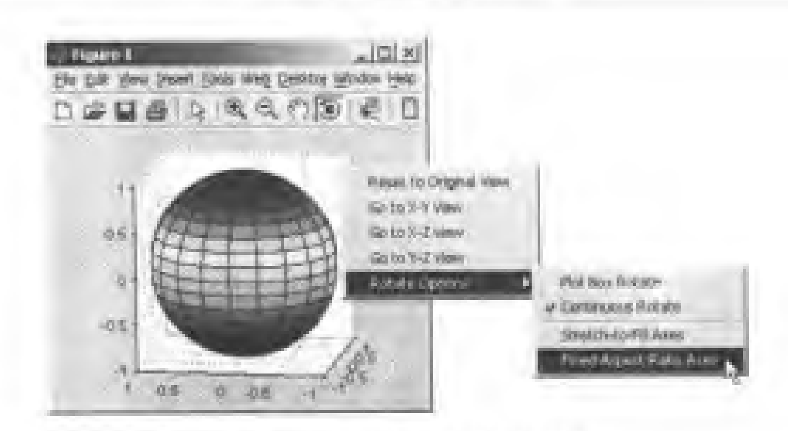



图 8-46　选择不同选项时的效果

### 1．基本曲线拟合

假设记录了等间隔数据 $t$ 对应的 $y$ 值，如下所示：

```
y = [-5.00 -5.74 -6.41 -5.67 -4.10 -1];
t = [0 .40 .80 1.20 1.60 2.00];
```

首先绘线形图，在命令窗口键入上面的命令行以后，用下面的语句绘图：

```
plot(t,y)
```

生成图 8-47。

然后通过在“Tools”菜单中选择“Basic Fitting”选项，显示基本拟合工具，如图 8-48 所示。采用三次多项式进行拟合。

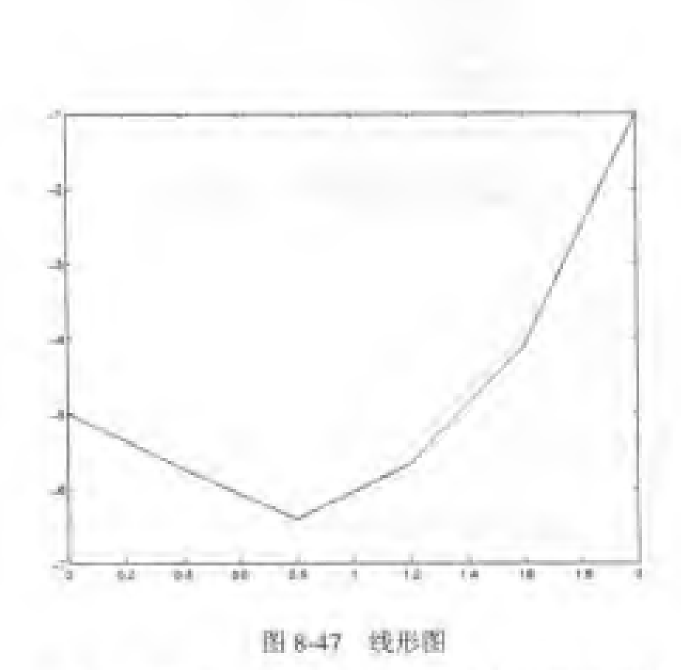


图 8-47　线形图




图 8-48　基本拟合工具

选择“cubic”选项以后，基本拟合工具用三次多项式拟合曲线，并在相同范围内显示多项式的图形，如图 8-49 所示。作为可选项，该工具还可以显示拟合曲线的多项式方程。

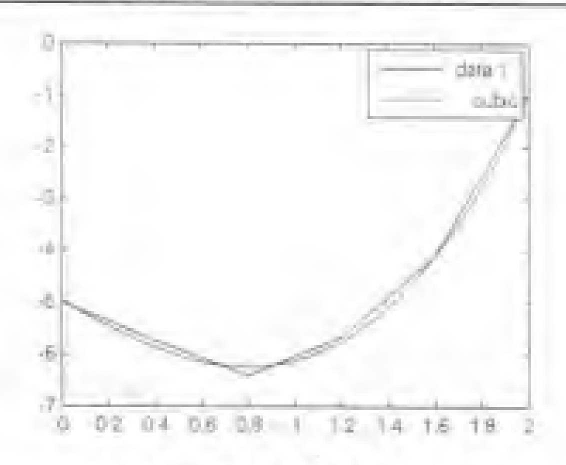



图 8-49　在线形图上叠加三次曲线

## 2. 在图形上添加基本统计图

MATLAB 数据统计工具计算与数据中心趋势和变异有关的基本统计量，然后在同一幅图中绘制这些统计量。

要获取统计量，从“Tools”菜单中选择“Data Statistics”选项。MATLAB 为图中的每个数据集计算一个统计量，并把结果显示在“Data Statistics”对话框中。

例如，图 8-50 直接在分布图中添加均值线。

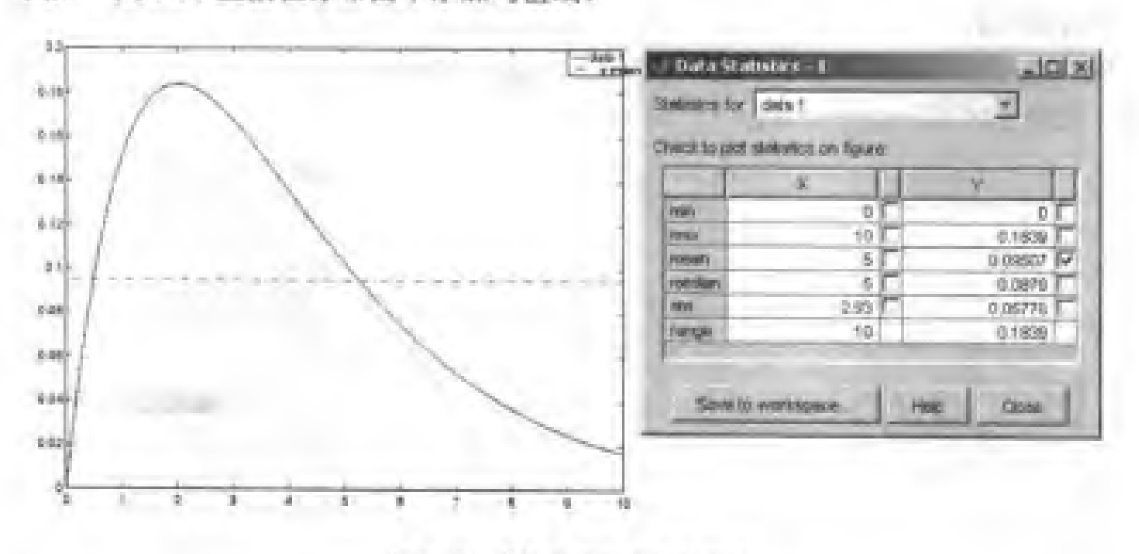



图 8-50　在分布图中添加均值线

## 3. 把统计量保存到工作空间

单击“Save to workspace”按钮，把统计量集合作为工作空间变量保存。数据统计工具把统计量作为结构保存。

下面的例子在图中添加数据集的均值线。按照下面的步骤进行操作：

（1）绘制数据。例如，使用下面的命令绘图。

```
load census
plot(cdate,pop,'+')
```

（2）从“Tools”菜单中选择“Data Statistics”选项。数据统计工具计算 x-data 和 y-data 的基本统计量，并把结果显示在对话框中。

（3）通过选择核选框来选择要绘在图中的统计量。例如，要在图中添加人口数据的均值图，单击数据后面的核选框，数据统计工具将在图中添加均值线，如图 8-51 所示。

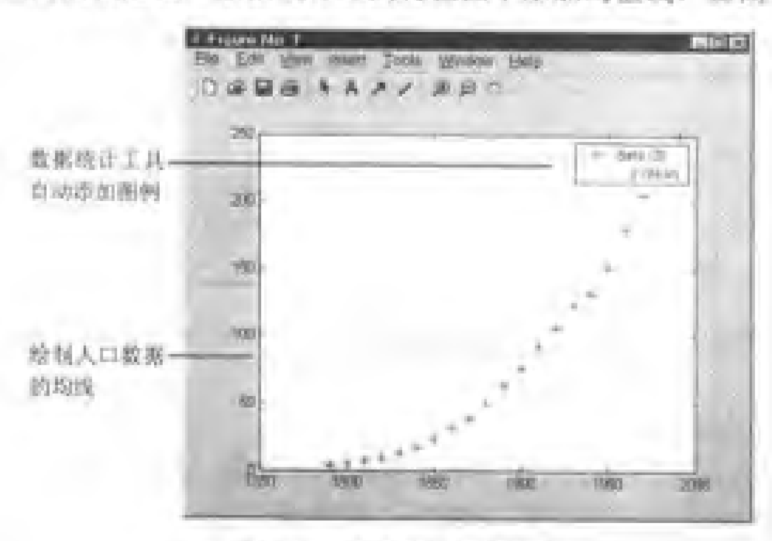



图 8-51 在图中添加均值线

### 4. 使用有数据统计量的图例

激活数据统计工具时，它计算绘图数据的统计量并自动添加一个图例，如图 8-52 中所示。




图 8-52 "Save Statistics to Workspace" 对话框

添加一个或多个统计量的图形到图中时，数据统计工具为新图形在图例中添加一个入口。数据统计工具会给绘图统计量指定一个名称，以在图例中进行识别。本例中，绘图统计量在图例中的名称为 y mean。

### 5. 格式化数据统计图

数据统计工具用颜色和线型来分辨图中的各种图元。这些属性都可以改变。

按照下面的步骤修改绘图统计量的属性：

（1）使图形窗口中的绘图编辑模式可用；

（2）双击统计量图形，启动属性编辑器。利用该编辑器获取用于绘制统计量的直线对象的属性，也可以通过在图上右击来获得这些属性的子集。该操作显示图形的上下文菜单，其中包括了设置图形属性的选项。

（3）改变图形的属性并单击 "Apply" 按钮。

### 6. 用数据统计工具绘制的统计量

注意，只能用数据统计工具生成二维数据（向量或矩阵）的统计量。

表 8-1 列出了用数据统计工具可以计算的统计量，其中包括了用于计算这些统计量的函数。

表 8-1　数据统计工具可以计算的统计量

| 统计量 | 描　　述 | MATLAB 函数 |
|---|---|---|
| 最大值 | 数据集中的最大值 | max |
| 最小值 | 数据集中的最小值 | min |
| 均值 | 数据集中所有值的平均值 | mean |
| 中值 | 数据集中所有值的中值 | median |
| 极差 | 数据集中最大值和最小值的差。数据统计工具不会绘制极差统计量 | n/a |
| 标准差 | 描述数据集中数据变异性的一个统计量。数据统计工具用两条线形图元表示数据集均值两侧标准差的边界 | std |

如果显示了数据统计量工具，并且改变图形的 x-data 或 y-data，数据统计工具会自动重新生成统计量。

**7．察看多幅图形的统计量**

数据统计工具为图中的每个二维图形计算基本统计量，但一次只能在一幅图中显示统计量。在图中察看特殊图形的统计量，按照下面的步骤进行。

（1）在数据统计对话框中单击“Statistics for”菜单，打开它。

该菜单列出了图中的所有数据集，通过它的 tag（与图形对象相连的自定义文本字符串）来识别每个数据集。对于没有 tag 的图形，数据统计工具用 data1 来识别第一幅图，用 data2 识别第二幅图，如此类推。

（2）从列表框中选择一幅图，数据统计工具会更新显示对话框中的值。

**8．将统计量保存到 MATLAB 工作空间**

按照下面的步骤将数据统计工具生成的统计量保存到工作空间（注意，必须对包含多个图元的图重复此过程）：

（1）单击“Save to workspace”按钮；

（2）在“Save Statistics to Workspace”对话框中指定要保存的统计量集合，x-data 或 y-data，并指定保存统计量的变量名称。

数据统计量将每个统计量集合保存在一个结构中，例如，将 x-data 的统计量保存在变量 census_dates 中，则该结构的内容可能会与下面的类似。

```
census_dates =
         min: 1790
         max: 1990
        mean: 1890
      median: 1890
         std: 62.0484
       range: 200
```

## 8.3　标注图形

用文本等标注图形可以增强图形传递信息的能力。本节介绍 MATLAB 中创建标注的技巧。

### 8.3.1　如何标注图形

MATLAB 提供了很多标注图形的特性，包括：

- 在图形窗口中添加文本、直线箭头、矩形、椭圆和其他标注对象；
- 将标注信息锚定在数据空间中；
- 添加图例和色条；
- 添加坐标系标签和标题；
- 编辑图形对象的属性。

**1．标注工具**

从“View”菜单中选择“Plot Edit Toolbar”选项，显示绘图编辑工具条，如图 8-53 中所示。

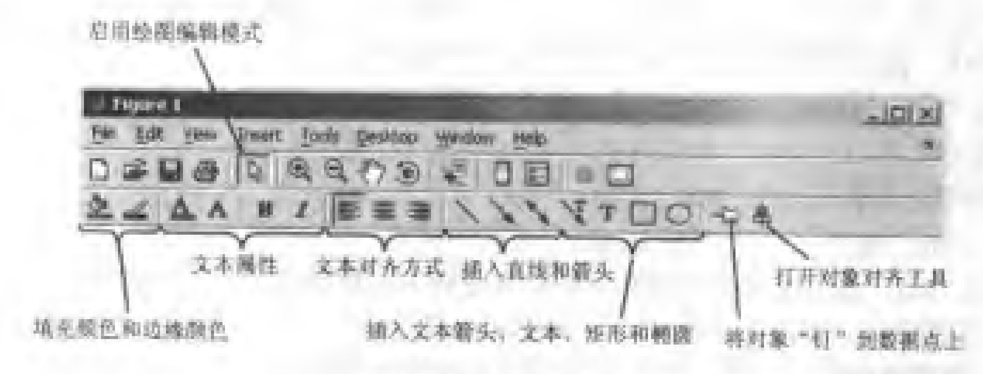



图 8-53　绘图编辑工具条

可以从图形面板中获取基本标注工具，从“View”菜单中选择“Figure Palette”选项，显示图形面板，如图 8-54 所示。

可以从“Insert”菜单中获取标注特性，如图 8-55 中所示。

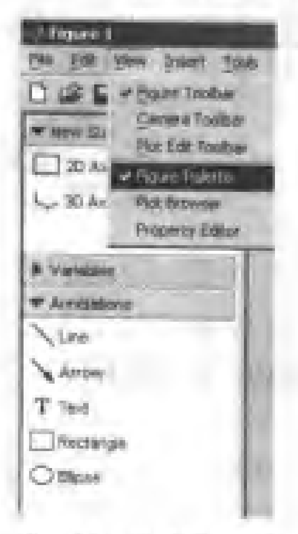



图 8-54　显示图形面板

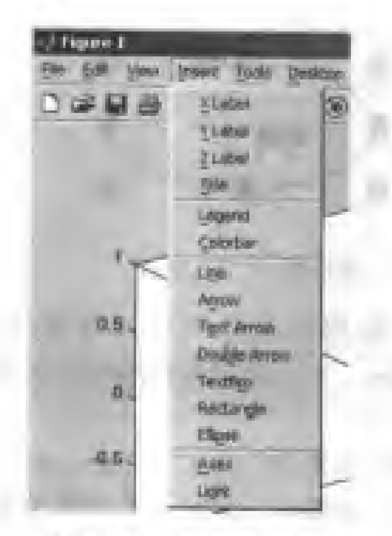



图 8-55　“Insert”菜单

**2．标注函数**

可以用 MATLAB 命令添加标注，表 8-2 列出了用于创建标注的函数。

表 8-2　创建标注的 MATLAB 函数

| 函　数 | 功 能 介 绍 |
|---|---|
| annotation | 创建直线、箭头、文本箭头、双头箭头、文本框、矩形和椭圆等标注 |
| xlabel, ylabel, zlabel | 向各个坐标轴添加文本标签 |
| title | 向图形添加标题 |
| colorbar | 向图形添加色条 |
| legend | 向图形添加图例 |

### 3. 将图形区域包围在矩形或椭圆中

可以用矩形或椭圆重点显示图形区域。当任何一个对象被选中时，可以移动和缩放它，并通过右击菜单中的选项修改它的行为和外观。

在绘图编辑工具条中单击▭◯，或者从“Insert”菜单中选择“Rectangle”或“Ellipse”选项，可以插入矩形或椭圆。此时，MATLAB 将鼠标光标变为十字丝形，表示可以通过单击、拖拉和释放鼠标来定义对象的大小和形状。

可以用“钉”操作将矩形固定在指定点上。有 3 种方法可以“钉”住矩形：

- 在矩形上右击，显示上下文菜单；
- 在图形工具条中选择“pin”图标；
- 从“Edit”菜单中选择“Pin Object”选项。

默认时，“钉”操作会将矩形或椭圆的左下角固定在指定点上。可以通过单击和拖拉移动固定点。拖拉时，鼠标光标变为钉形，如图 8-56 中所示。

注意，矩形或椭圆被钉住以后就不能再进行拖拉和缩放操作。

在矩形或椭圆上单击鼠标右键，显示上下文菜单，如图 8-57 所示。

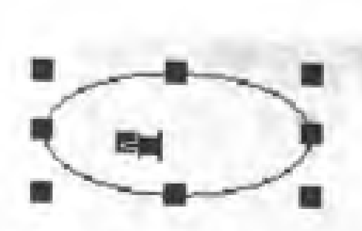


图 8-56　“钉”操作

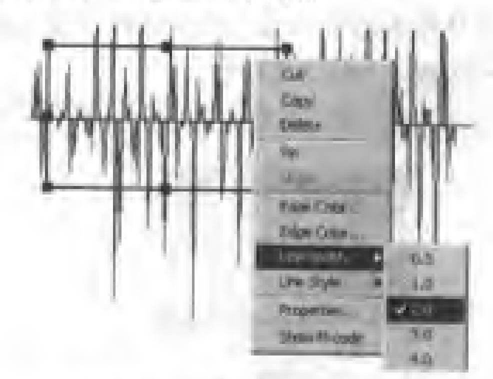


图 8-57　用菜单选项编辑矩形

可以利用该菜单中的选项修改矩形或椭圆，也可以通过从上下文菜单中选择“Properties...”选项用属性编辑器来设置矩形和椭圆的属性。属性编辑器显示上面描述的相同属性。

在属性编辑器中单击“Inspector”按钮，显示属性查看器。属性查看器为选中的标注对象显示所有属性。但是，不应该改变其中的某些属性，因为这样做会影响标注对象的正确功能。

### 4. 文本框标注

文本框是一个包含多行文本的矩形，可以将文本框放在图形窗口中的任意位置。

在工具条上单击 T，然后在要放置文本字符串的地方单击鼠标，可以添加文本框。创建文本框以后，可以改变文本框的大小和进行拖拉操作。也可以从“Insert”菜单中选择“TextBox”选项。

文本框的选择行为与其他标注对象的不同：

- 删除文本框时，单击并选择它；
- 编辑文本框时，双击文本框内部；
- 右击文本框并在弹出的上下文菜单中选择“Properties...”选项，显示文本框的属性。

可以使用“钉”操作将文本固定在某个特定的点上。可以有三种方法：

- 在文本框中右击鼠标，利用上下文菜单进行设置；
- 在图形工具条上选择“pin”图标；
- 在“Edit”菜单中选择“Pin Object”选项。

默认时，MATLAB 会将文本框的左下角固定在指定点上，可以通过单击和拖拉移动固定点。拖拉时，鼠标光标变为钉形，如图 8-58 中所示。

右击文本框显示它的上下文菜单，利用它对文本框进行一系列操作。图 8-59 中，文本框的背景色用上下文菜单设置为黄色。

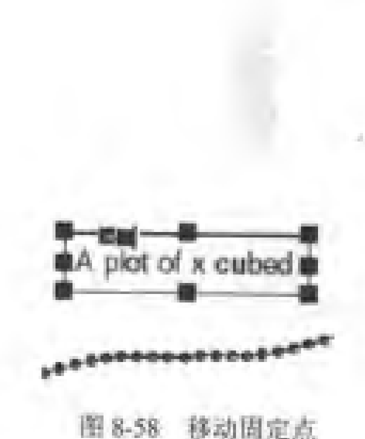



图 8-58 移动固定点

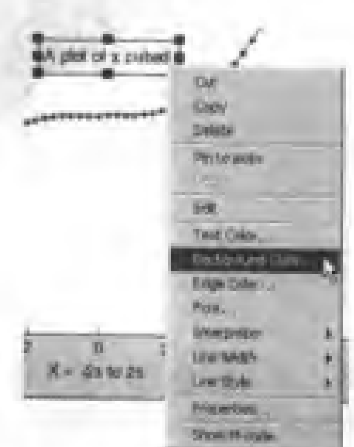



图 8-59 用上下文菜单编辑文本框

可以通过从上下文菜单中选择“Properties...”选项设置文本框属性。它显示了上面描述的相同属性。单击属性编辑器上的“Inspector”按钮，显示属性查看器。属性查看器显示所有的文本框属性。

**5. 标注线和箭头**

可以给图形添加直线和 3 种类型的箭头，并将它们放到图中任何位置。这 3 种类型的箭头包括：

- 单头箭头；
- 有标注文本框的箭头（文本箭头）；
- 双头箭头。

在图形窗口的工具条上单击 中的合适按钮，然后在图中单击、拖拉和释放鼠标，绘制直线和箭头。




图 8-60　文本箭头

文本箭头是文本和箭头的组合，它对于标注图中的点很有用。在工具条中单击箭头上有 T 字符的按钮，如图 8-60 所示，在图中插入文本箭头并在方框中键入文本。

可将箭头末端固定在图中的目标点上，并让文本框在放大和平移时自动设置新位置。有 3 种方法可以进行此项操作：

- 在对象上单击鼠标右键，然后在弹出的上下文菜单中单击“Pin”选项；
- 在绘图编辑工具条中选择“pin”图标；
- 在“Edit”菜单中选择“Pin Object”选项。

**6．在图中添加色条**

色条显示当前的颜色映射表，并指示数据值向颜色的映射。图 8-61 显示了有二维等值线图的表面图。注意右侧的色条如何表示 z 轴数据值与表面图和等值线图之间的对应关系。

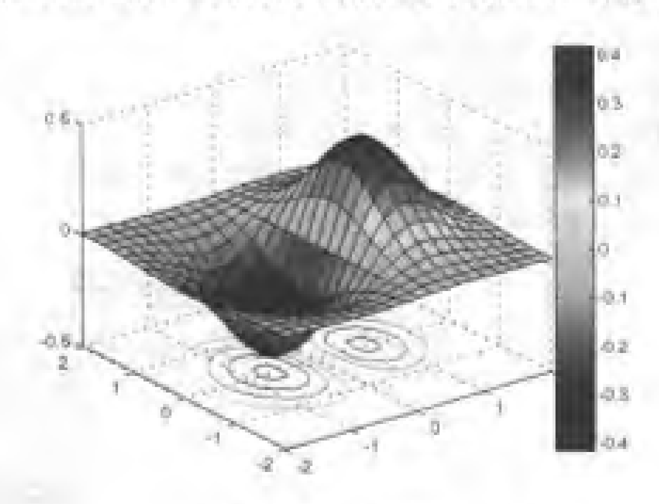


图 8-61　添加色条

在工具条上单击按钮，或者通过从“Insert”菜单中选择“Colorbar”选项来添加色条。绘图编辑模式可用时，可以选择、移动和缩放色条。也可以用 colorbar 命令在图中添加色条。

有几种方法可以在同一图形窗口中重新定位色条：

- 启用绘图编辑模式，然后选择色条并把它拖拉到目标位置；
- 右击色条，显示它的上下文菜单。将鼠标移向“Locations”选项，并选择一个预定义的色条位置；
- 在色条上右击，显示上下文菜单，并选择“Properties...”选项。此操作显示属性编辑器。

**7．给图形添加图例**

图例提供了图上不同数据的控制键。图 8-62 显示了 4 个函数图形的图例。

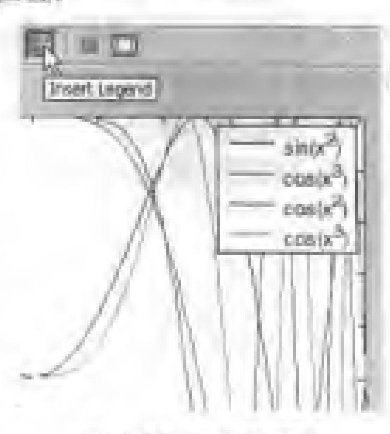



图 8-62　函数的图例

在工具条上单击按钮，或者在“Insert”菜单上选择“Legend”选项，可以添加图例。绘图编辑模式可用时，可以选择、移动和缩放图例，也可以用 legend 命令在图中

添加图例。

默认时，图例用字符串 data1,data2 等标注每个图形对象，可以通过双击此文本来改变它。在编辑模式下，可以重新键入文本字符串。

定位图例有下面几种办法：

- 使绘图编辑模式可用，选择图例，然后把它拖拉到目标位置；
- 右击图例并显示它的上下文菜单，将鼠标放在"Location"选项上，从子菜单中选择一个预定义位置；
- 在图例上右击，显示它的上下文菜单并选择"Properties..."选项，显示属性编辑器；也可以选择图例的垂向或水平向显示方位，用"Orientation"选项作此选择。

8. 固定对象

使用"钉"操作可以将对象固定到图形中的指定点上。使用该操作可以在平移或缩放图形时保持标注始终连接两个相同的点。例如，图 8-63 中用双头箭头指出了两个图形中相同位置的点。

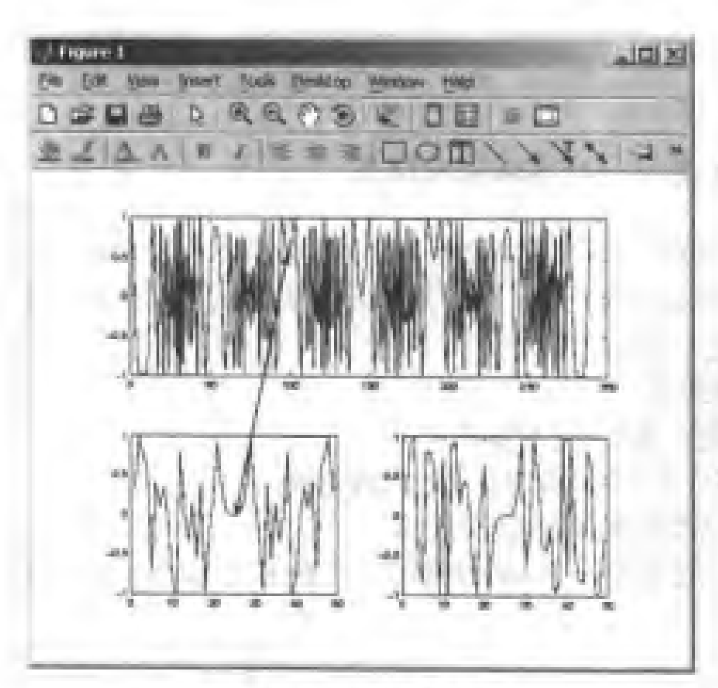


图 8-63 确定图中点的位置

如果在上面的坐标系中进行水平缩放，然后平移图形，显示数据前 120s 的图形，双头箭头始终连接图中的同一位置，如图 8-64 所示。

固定对象，需要通过单击图标或选择"Edit"菜单中的"Pin Object"选项来启用固定模式，然后单击要固定的点。

要取消固定，右击图形，然后在弹出的上下文菜单中选择"Unpin"选项。

可以固定标注线、箭头、矩形、椭圆和文本框。

本模式启用时，坐标系、矩形、箭头和直线的左上角自动与网格对齐。移动和缩放这些对象时，大小和位置跳向下一个网格位置。

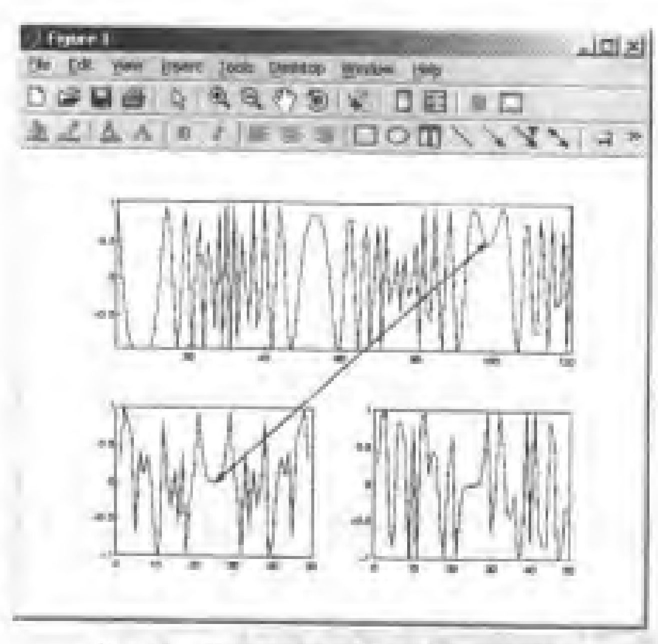


图 8-64　移动图形时箭头始终连接相同点

## 8.3.2　对齐工具——对齐和分布对象

对齐工具如图 8-65 所示。使用对齐工具，可以确定选中对象的相对位置。指定的对齐/分布操作适用于所有选中的元素。单击图标或从“Tools”菜单中选择“Align Distribute Tool”选项，可以显示对齐工具。

**1．对齐工具的功能**

对齐工具提供两种类型的定位操作：

- Align　将所有选中的对象与一条参考线对齐；
- Distribute　均匀间隔所有选定的对象。

下面的例子演示如何用 3 个对应的坐标系对齐 3 个文本框。图 8-66 中显示初始情况下的图形。




图 8-65　对齐工具

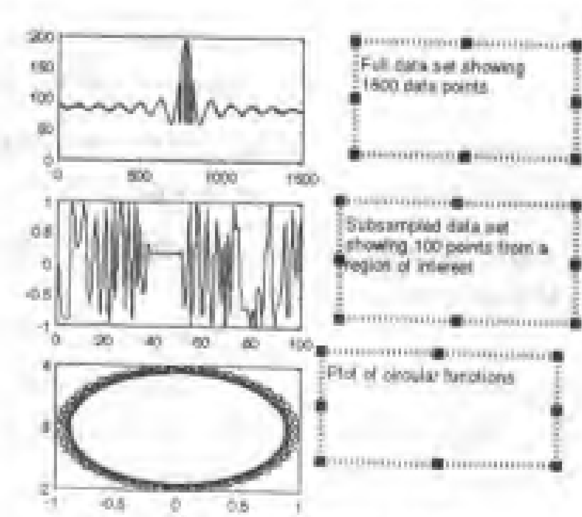



图 8-66　初始情况下的图形

按住 Shift 键的同时选择全部 3 个文本框，然后如图 8-67 中设置对齐工具，设置选项如下：

- 将垂向间距设置为 40 像素；
- 将水平对齐方式设置为右对齐；
- 单击 “Apply” 按钮。

图 8-66 设置后的效果如图 8-68 所示。




图 8-67　设置选项

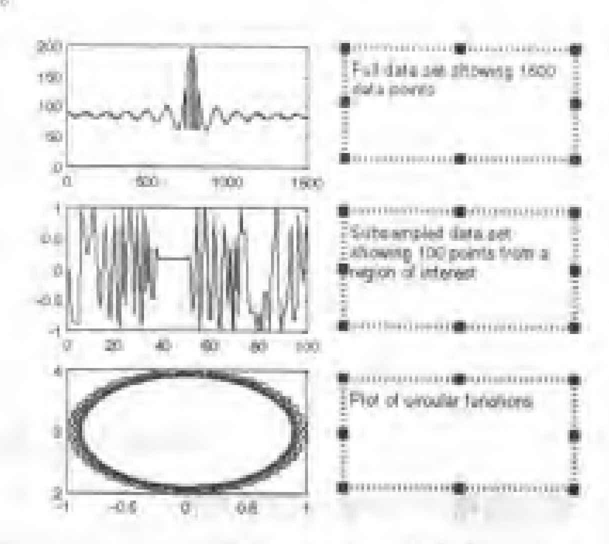



图 8-68　设置效果

“Tools” 菜单包含对齐和分布选项，如图 8-69 所示。

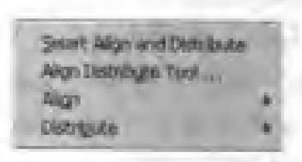



图 8-69　“Tools” 菜单中的对齐和分布选项

“Smart Align and Distribute” 选项把等间距间隔的对象按行或列对齐。很多对象需要对齐时将它们定位到 $m \times n$ 的网格。例如，图 8-70 中包含 6 个坐标系，它们近似排列成两列。

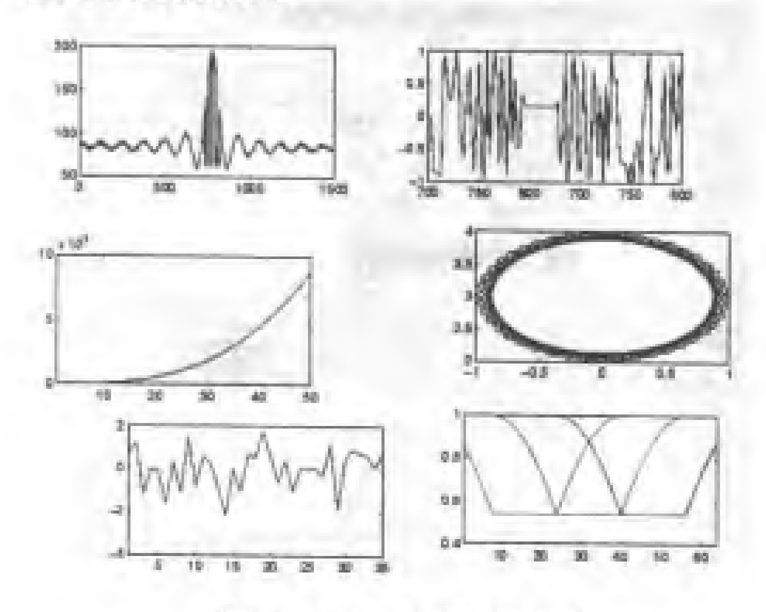


图 8-70　准备对齐的 6 个坐标系

要在坐标系中对齐所有坐标系，选择每个坐标系，然后从“Tools”菜单中选择“Smart Align and Distribute”选项，结果如图 8-71 所示。

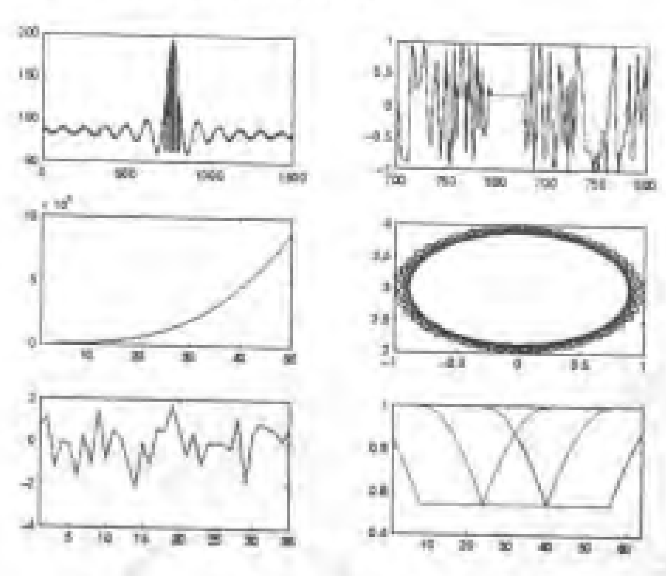


图 8-71　对齐后的效果

**2．跳至网格——对象与网格对齐**

图形窗口可以显示网格，网格是定位对象的辅助工具之一。结合网格，还可以利用一种称为“跳至网格”的技术，使对象移动时自动与网格对齐。

在“Tools”菜单上选择“View Layout Grid”选项，可以在图形窗口背景上显示网格。在“Tools”菜单上选择“Snap To Layout Grid”选项，可以迫使对象与网格对齐。从“Tools”菜单中选择“Edit Plot”选项，使绘图编辑模式可用，可以移动对象。

图 8-72 演示了有 4 个子图的图形窗口。可以选择这 4 个坐标系中的任何一个并进行移动。注意，所有坐标轴标签和标题会跟着一起移动，而标注对象的移动与坐标系无关。

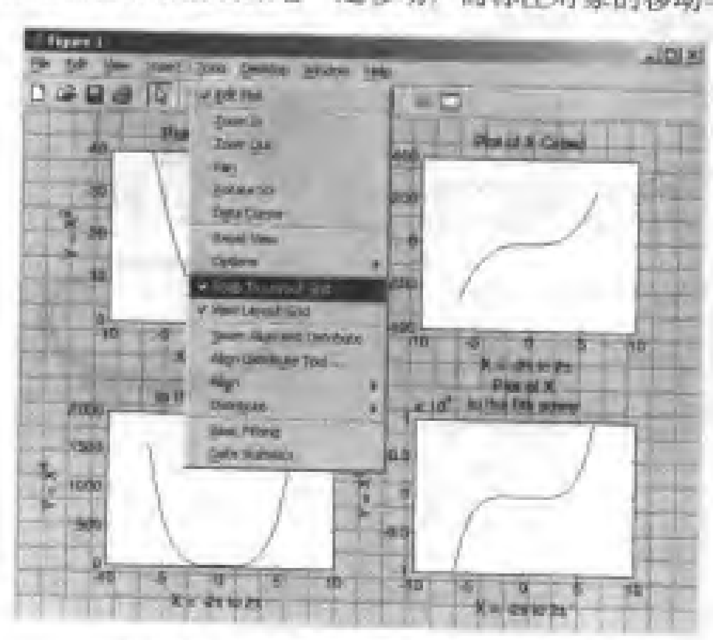


图 8-72　通过设置将 4 个子图与网格对齐

### 8.3.3　添加标题

在 MATLAB 中，标题是坐标系顶部的文本字符串。一般情况下，标题定义图形的主题，如图 8-73 中所示。

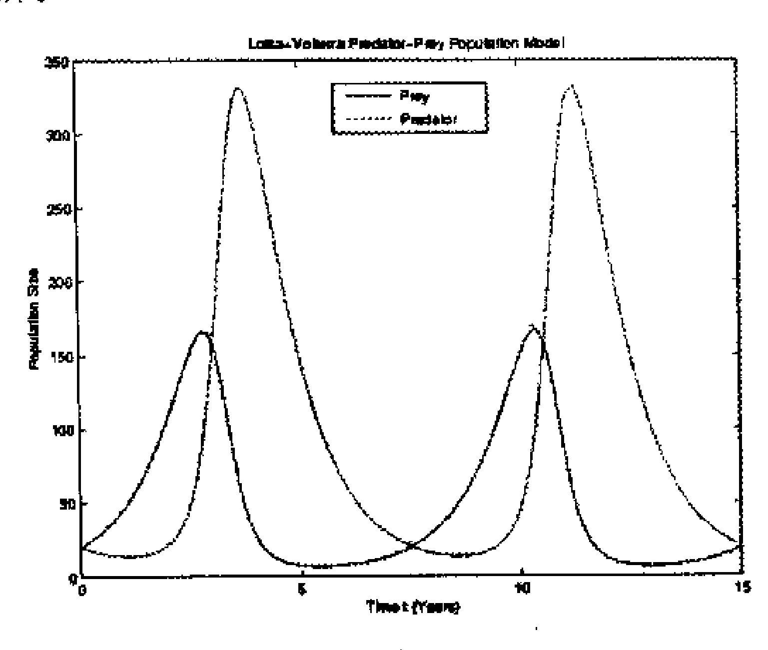


图 8-73　给图形添加标题

### 8.3.4　添加坐标系标签

在 MATLAB 中，坐标系标签是与图形中 $x$, $y$ 或 $z$ 轴对齐的文本字符串，坐标轴标签可以帮助解释每个坐标轴所对应的变量意义和度量单位。在图中添加坐标系标签大致有下面几种方法。

（1）利用"Insert"菜单中的"Label"选项

- 单击"Insert"菜单并选择与要标注的坐标轴对应的标签选项，包括"X Label"，"Y Label"和"Z Label"。MATLAB 会沿坐标轴或在已经存在的坐标轴标签周围显示一个文本入口框。
- 输入标签的文本，或编辑已经存在的标签的文本。
- 在背景中单击任何位置，关闭标签周围的文本入口框。如果单击图中的另一个对象，例如坐标轴或直线，会在关闭标签文本入口框的同时自动选择所单击的对象。

（2）使用属性编辑器添加坐标系标签（见图 8-74）

- 在"Tools"菜单中单击"Edit Plot"选项，启用绘图编辑模式。
- 在图中双击坐标系，启用属性编辑器，也可以通过双击坐标系或从上下文菜单中选择"Properties..."选项来启用属性编辑器。
- 根据要添加的坐标轴标签选择"X Axis"，"Y Axis"和"Z Axis"面板，在文本入口框中输入标记文本。



图 8-74　用属性编辑器添加坐标系标签

可以用属性编辑器旋转坐标轴标签：

● 在“Tools”菜单中选择“Edit Plot”选项，启用绘图编辑模式；

● 通过选择要旋转的坐标轴标签来显示属性编辑器。右击选中的文本，然后从上下文菜单中选择“Properties...”选项；

● 单击“Inpector”按钮，显示属性查看器，如图 8-75 所示；

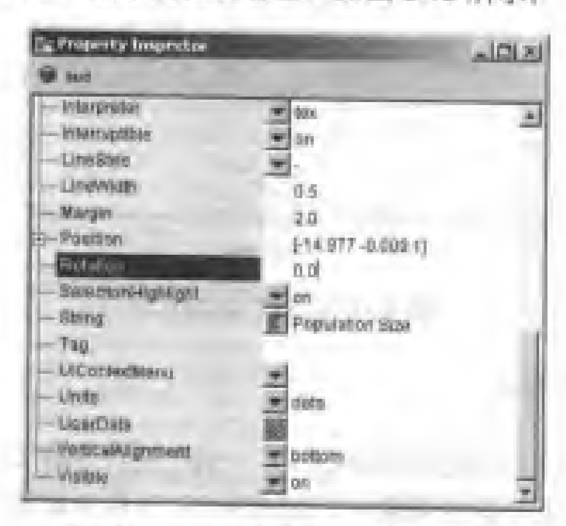


图 8-75　用属性编辑器旋转坐标轴标签

● 选择“Rotation”属性，值设置为 0；

● 用左键单击选中的标签，把文本拖拉到指定的位置并释放鼠标。

可以用 xlabel,ylabel 和 zlabel 命令添加 $x$, $y$ 和 $z$ 轴的标签。例如，下面的语句标注坐标轴并添加标题。

```
xlabel('t = 0 to 2\pi','FontSize',16)
ylabel('sin(t)','FontSize',16)
title('\it{Value of the Sine from Zero to Two Pi}','FontSize',16)
```

生成图 8-76。

标签命令会自动确定文本字符串的位置。

坐标轴标记是文本对象，可以通过给对象的 Rotation 属性指定值来旋转它。$x$, $y$ 和 $z$ 轴标签的句柄分别保存在 XLabel,YLabel 和 ZLabel 属性中。按照下面的步骤旋转 $y$ 轴标签，使得文本水平：

● 用 YLabel 属性获取文本对象的句柄；

● 将 Rotation 属性的值设置为 0.0 度。

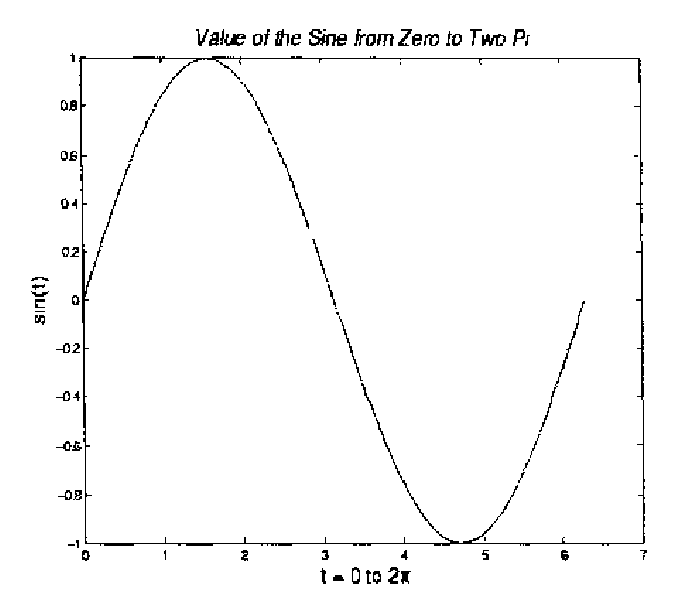



图 8-76　在图中添加标题和坐标轴标签

例如，下面的语句旋转当前坐标系上的 y 轴标签：

```
set(get(gca,'YLabel'),'Rotation',0.0)
```

可以通过拖拉文本来重置坐标轴标签。按照下面的步骤进行：

- 从“Tools”菜单中选择“Edit Plot”选项，启用绘图编辑模式；
- 选择要重置的标签文本；
- 用左键在选定的标签上单击，把文本拖拉到目标位置并释放鼠标。

### 8.3.5　添加文本标注

可以在图中添加自由形式的文本，帮助解释数据。

如果启用绘图编辑模式，可以通过单击图形中的区域或图形背景并输入文本创建文本标注。也可以从命令行添加文本标注，使用 text 或 gtext 命令。

注意，使用绘图编辑模式或 gtext 命令使得在图中任何位置上放置文本标注更容易。把文本标注放在数据集中的指定点上时，使用 text 命令。

**1. 用 text 或 gtext 命令创建文本标注**

要用 text 函数创建文本标注，必须用 x 和 y 坐标在图中指定文本及其位置。用鼠标在数据空间的指定点上放置文本标注时使用 gtext 命令。

例如，下面的代码在指定位置上创建文本标注。

```
str1(1) = {'Many Predators;'};
str1(2) = {'Prey Population'};
str1(3) = {'Will Decline'};
text(7,220,str1)

str2(1) = {'Few Predators;'};
str2(2) = {'Prey Population'};
str2(3) = {'Will Increase'};
```

```
text(5.5,125,str2)
```

本例还演示了如何用单元数组创建多行文本。

### 2. 计算文本标注的位置

也可以计算图中文本标注的位置。下面的代码在 3 个数据点上添加标注。

```
text(3*pi/4,sin(3*pi/4),...
    '\leftarrowsin(t) = .707',...
    'FontSize',16)

text(pi,sin(pi),'\leftarrowsin(t) = 0',...
    'FontSize',16)

text(5*pi/4,sin(5*pi/4),'sin(t) = -.707\rightarrow',...
    'HorizontalAlignment','right',...
    'FontSize',16)
```

文本字符串“sin(t)=-.707\rightarrow”的 HorizontalAlighment 属性值设置为 right，把该字符串放在点[5*pi/4,sin(5*pi/4)]的左侧。图形效果如图 8-77 所示。

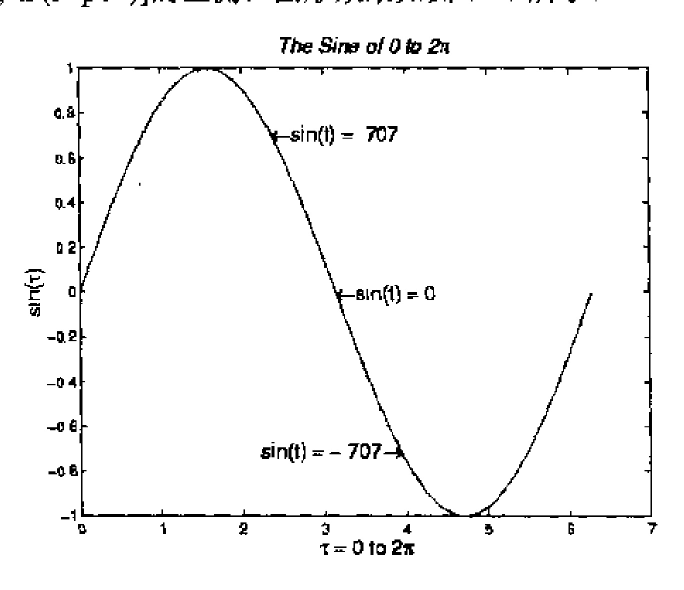



图 8-77　在图中添加标注

可以用文本对象在任意位置标注坐标轴。例如，下面绘函数 $y = Ae^{-\alpha t}$ 的图形，其中，$A$=0.25, $\alpha$ =0.005, $t$=0~900。

```
t = 0:900;
plot(t,0.25*exp(-0.005*t))
```

为了标注 $t$=300 处的点，用绘图函数计算文本的坐标。

```
text(300,.25*exp(-0.005*300),...
'\bullet\leftarrow\fontname{times}0.25{\ite}^{-0.005{\itt}} at {\itt} =
300',...
'FontSize',14)
```

生成图形如图 8-78 所示。

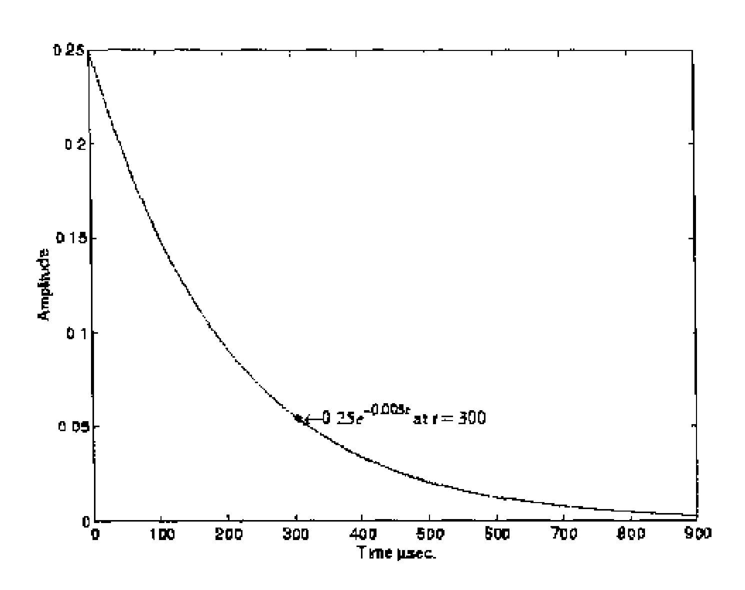



图 8-78　在指定位置添加标注

定义文本 Position 属性的表达式为

$$x = 300, y = 0.25e^{-0.005\times300}$$

HorizontalAlighment 和 VerticalAlighment 属性控制文本字符相对于指定的 $x, y$ 和 $z$ 坐标的距离。默认时

```
HorizontalAlignment = left
VerticalAlignment = middle
```

MATLAB 不会精确地把文本 String 放在指定位置 Position 上，例如，前面显示了一个用文本标注点的图形。放大图形可以察看文本的实际位置，如图 8-79 所示。

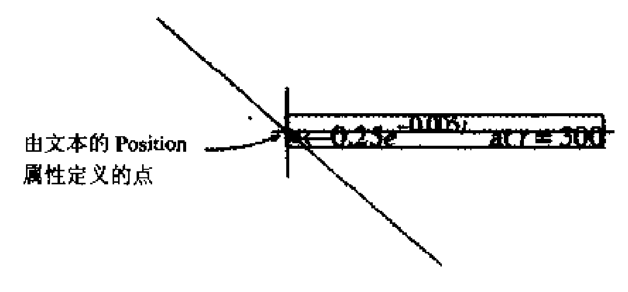



图 8-79　文本的实际位置

如图 8-79 中所示，小点表示文本的 Position 属性指定的点。大点表示文本的 String 属性中第一个字符定义的点。

假设要用文本标注图上的最大值和最小值。下面用 peaks 矩阵中的一列数据绘图，然后用绘图函数确定文本的位置和要显示的内容，进行标注。

```
Z = peaks;
h = plot(Z(:,33));
x = get(h,'XData'); % 获取绘图数据
y = get(h,'YData');
imin = find(min(y) == y);% 找到最小值和最大值的索引号
imax = find(max(y) == y);
```

```
text(x(imin),y(imin),[' Minimum = ',num2str(y(imin))],...
      'VerticalAlignment','middle',...
      'HorizontalAlignment','left',...
      'FontSize',14)
text(x(imax),y(imax),['Maximum =   ',num2str(y(imax))],...
      'VerticalAlignment','bottom',...
      'HorizontalAlignment','right',...
      'FontSize',14)
```

生成图形如图 8-80 所示。

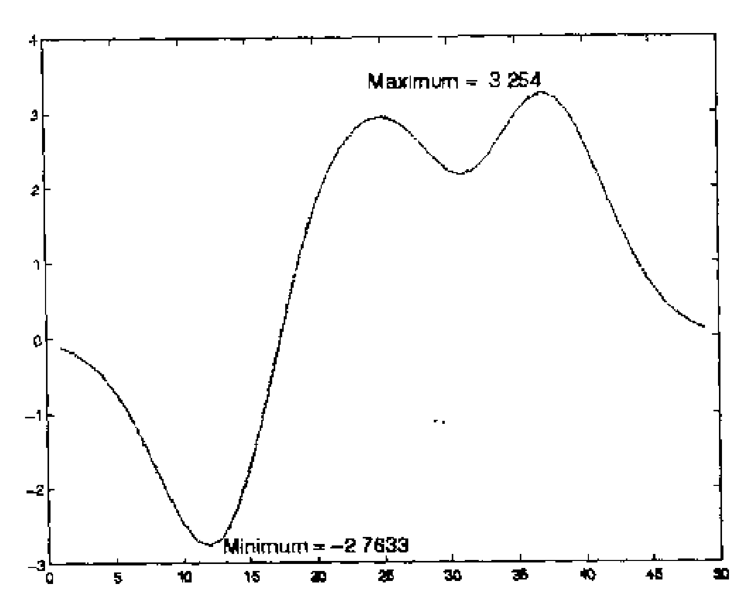



图 8-80　标注图中的最大值和最小值

与对齐属性一致，text 函数相对于坐标指定的点来定位字符串。对于最小值，字符串显示在文本位置点的右侧；对于最大值，字符串显示在文本位置点的上面或左侧。文本总是保持在计算机屏幕平面上，与视图设置无关。

**3．编辑文本对象**

可以在图中编辑任何标签和标注，按下面的步骤进行：

- 启用绘图编辑模式；
- 在字符串上双击，或右击字符串以后从上下文菜单中选择“String”选项，在文本周围显示一个编辑框；
- 改变文本；
- 在文本编辑框外任意一处单击鼠标，终止编辑。

**4．数学标记、希腊字母和 TEX 字符**

使用 TEX 类型的字符序列可以包含数学标记和希腊字母。下面的例子用 TEX 字符序列创建图形标签。假设已经有一幅线形图，下面的语句在线形图中添加标题和 $x, y$ 轴标签。

```
title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
xlabel('Time \musec.')
ylabel('Amplitude')
```

绘图效果如图 8-81 中所示。

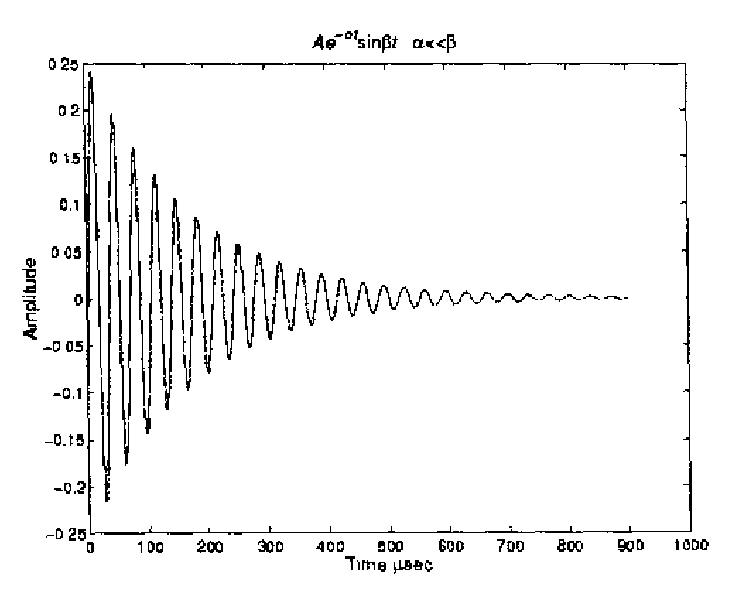



图 8-81　绘图效果

任何字符串变量都可以指定为文本 String 属性的值。下面演示如何将矩阵、单元数组和数值变量作为 text 函数的变量。

（1）字符变量

例如，矩阵 PersonalData 的每一行都包含了与个人有关的特定信息。

```
PersonalData = ['Jack Straw   ';'489 Main St.';'Wichita KN '];
```

要显示这些数据，索引目标行：

```
text(x1,y1,['Name: ',PersonalData(1,:)])
text(x2,y2,['Address: ',PersonalData(2,:)])
text(x3,y3,['City and State: ',PersonalData(3,:)])
```

（2）单元数组

使用单元数组，可以用一个单独的文本对象创建多行文本。每个单元的字符数不必相同。例如，下面的语句：

```
key(1)={'{\itAe}^{-\alpha\itt}sin\beta{\itt}'};
key(2)={'Time in \musec'};
key(3)={'Amplitude in volts'};
text(x,y,key)
```

生成图 8-82 所示的输出。

$Ae^{-\alpha t}\sin\beta t$

Time in μsec

Amplitude in volts

图 8-82　用单元数组创建多行文本

（3）数值变量

可以使用 num2str 函数，用文本字符串指定数值变量。例如，在命令行中键入

```
x = 21;
['Today is the ',num2str(x),'st day.']
```

MATLAB 会把这 3 个单独的字符串聚合成一个，即

```
Today is the 21st day.
```

因为结果是合法的字符串，可以把它作为文本 String 属性的值指定。

```
text(xcoord,ycoord,['Today is the ',num2str(x),'st day.'])
```

### 8.3.6 添加箭头和直线

绘图编辑模式可用时，可以在图形窗口中的任意位置添加箭头和直线，如图 8-83 中所示。也可以用箭头字符，利用 text 命令创建箭头。但是，这样创建的箭头只能在水平方向上指向左或指向右。

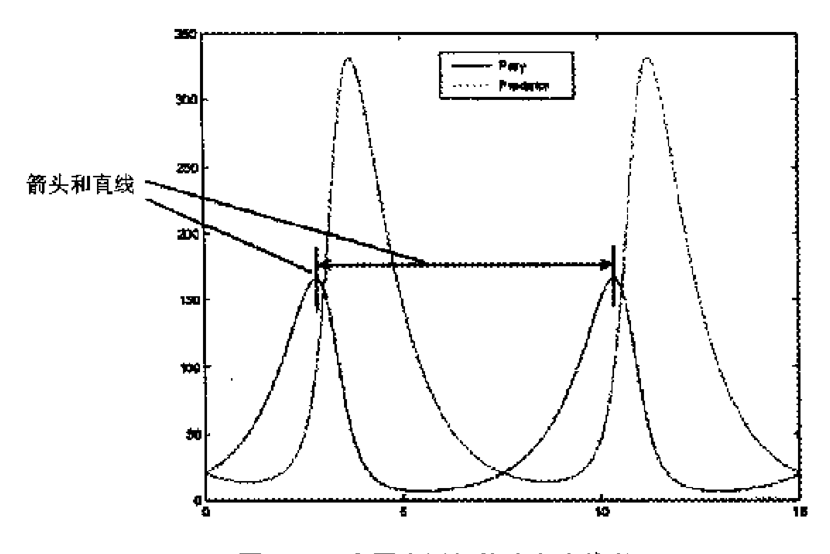



图 8-83 在图上添加箭头和直线段

按照下面的步骤在图中添加箭头和直线：

（1）单击“Insert”菜单并选择“Arrow”或“Line”选项，或在绘图编辑工具条中单击“Arrow”或“Line”按钮，MATLAB 将光标改为十字丝形；

（2）把光标放在图中直线或箭头的起点位置上，按下鼠标键，然后进行拖拉操作；

（3）释放鼠标。

可以用上下文菜单编辑箭头和直线标注的外观。

# 第 9 章　句柄图形对象

句柄图形对象是 MATLAB 绘图的基础，是构筑 MATLAB 图形这座大厦的砖和瓦。MATLAB 提供的所有二维、三维绘图工具都是利用这些基本对象建立起来的。学习句柄图形对象的重要性在于，可以深入到 MATLAB 图形编程的底层，可以具备自己创建专业图形的能力，甚至，如果有必要，可以创建自己的等值线图、向量图、曲面图、等值面图等。

相对于以前的各个版本，MATLAB 7.0 中句柄图形对象的结构和元素作了较大的调整。增加了部分对象，对各种图元对象和场景对象进行了重新分类。

## 9.1　面向对象的思维方式

面向对象是一种程序设计方法，是相对于面向过程而言的。所谓对象，是客观存在的事物或关系。比如笔是对象，纸是对象，几何图形也是对象。对象有特征，包括与其他对象相同的特征和不同的特征，这些特征称为属性；对象可以有行为，称为方法。如小猫这个对象有胡须这个属性，有跑、跳这样一些方法。

面向对象的优越性在于，可以重复使用对象进行编程，而且面向对象还可以描述继承、多态这样一些对象关系。相对于过程而言，对象是一个更为稳定的叙述单元。因为过程可能经常变化，稍有变化就不能直接重复调用这个过程；而对象更为稳定，比如猫有胡须，会跑、会跳，小时候是这样，老了也是这样，白猫是这样，黑猫还是这样。由于面向对象有这样一些优越性，它目前是主流的编程技术。

## 9.2　句柄图形对象的组织

### 9.2.1　句柄图形对象的层次结构

MATLAB 7.0 中句柄图形对象按层次体系组织，如图 9-1 所示。

前面讲了，MATLAB 7.0 中句柄图形对象的元素和组织发生了较大的改变。图 9-2 中是 MATLAB 6.x 版本中使用的句柄图形对象的层次结构。比较图 9-1 和图 9-2 可以发现，不同版本所使用的句柄图形对象主要有下面一些特点：

- 都有 4 个对象层次，Root 对象和 Figure 对象都作为第一层次和第二层次；
- 原来第四层次中的图元对象被放到核心对象节点中；
- 对象类型进一步丰富，增加了绘图对象、组对象、隐藏的 Annotation Axes 对象和 Annotation 对象。

对象的每一个实例都与名为句柄的惟一标识符相连。使用这个句柄，可以操作已经存在的图形对象，也可以在创建图形对象时给属性指定值。

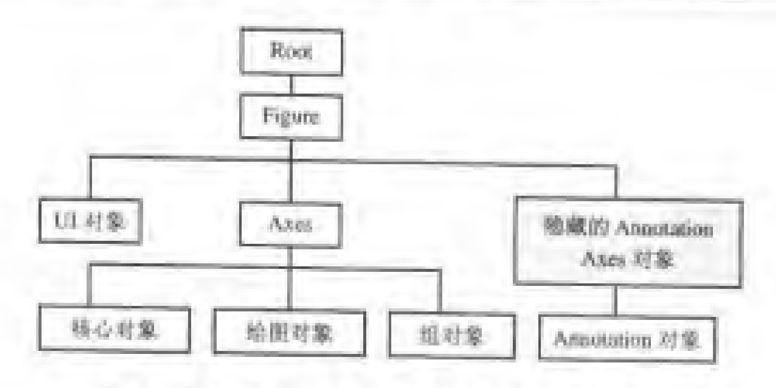



图 9-1　MATLAB 7.0 中句柄图形对象的层次体系

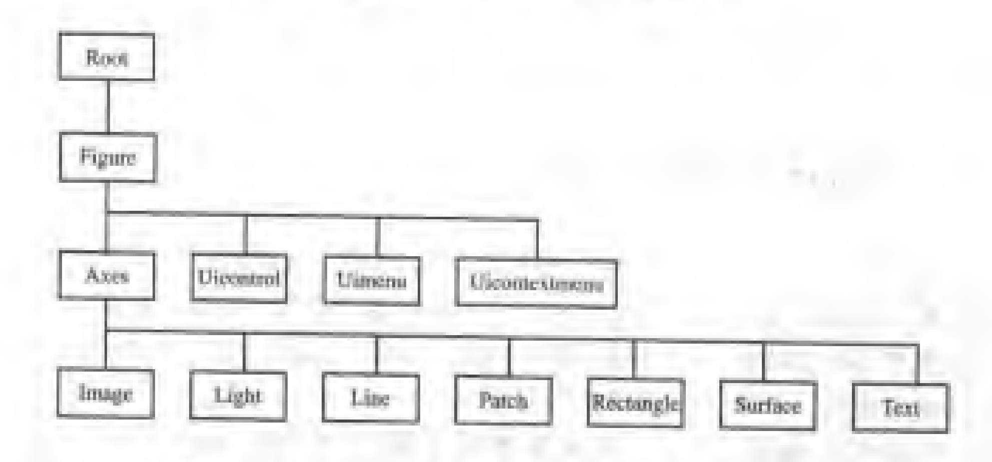



图 9-2　MATLAB 6.x 中句柄图形对象的层次结构

句柄图形对象的层次关系依赖于不同图形对象之间的相互关系。例如，绘制直线对象时，MATLAB 需要一个坐标系对象来定向，并提供参考方框。坐标系又需要一个图形窗口来显示坐标轴和它的子对象。

### 9.2.2　句柄图形对象的类型

MATLAB 7.0 中有两个基本的图形对象类型，即

- 核心图形对象　用于高级函数和组合对象；
- 组合对象　由核心图形对象组成，已经组合在一起，并提供了更便捷的接口。

组合对象组成了图形对象子类的基础：

- 绘图对象　该对象由基本图形对象组成；
- 标注对象　该对象存在于一个与其他图形对象分离的层上；
- 组对象　创建在某方面有共性的对象组，可以作为任何坐标系子对象的父对象，包括其他组对象；
- UI 对象　用于创建图形用户界面。

图形对象是独立的，所以通常图形显示包括很多对象，它们组合在一起，形成了有意义的图形。

## 9.3　图形窗口——Figure 对象

Figure 对象是 MATLAB 显示图形的窗口。图形窗口包括菜单、工具条、用户界面对象、上下文菜单、坐标轴和坐标轴子对象等。MATLAB 本身对于图形窗口的个数没有限制。

Figure 对象主要有两个方面的用途：

- 包含数据的图形；
- 包含图形用户界面。

Figure 对象可以同时包含图形和 GUI 组件。例如，一个利用数据绘图的 GUI 可以包含坐标系和用户界面对象，即 Axes 对象和 UI 对象。

图 9-3 显示了 Figure 对象可以包含的对象类型。

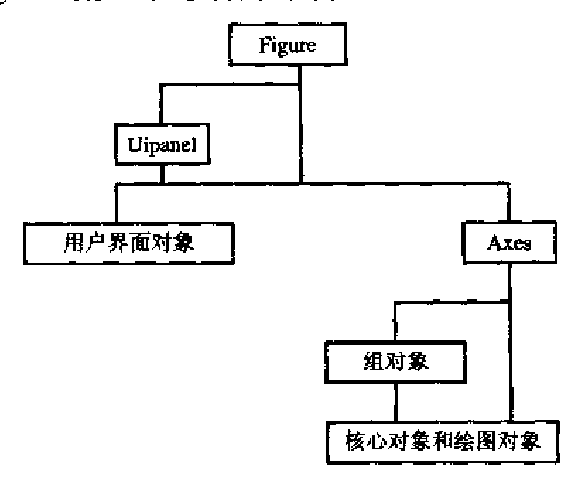



图 9-3　Figure 对象包含的对象类型

注意：Figure 对象和 Axes 对象都可以包含作为容器的子对象。用户界面面板可以包含 UI 对象和作为 Figure 对象的子对象，组对象可以包含坐标系子对象和作为 Axes 对象的子对象。

### 9.3.1　用于绘图的图形窗口

用 MATLAB 函数绘图时，如果先前没有图形窗口，会自动创建一个。如果打开了多个图形窗口，在当前窗口中绘图。

gcf 命令返回当前图形窗口的句柄或创建新的图形窗口。例如，可以用下面的命令查看一系列图形属性。

```
get(gcf)
```

根对象的 CurrentFigure 属性返回当前图形窗口的句柄。如果当前没有打开图形窗口，返回空值。例如，

```
get(0,'CurrentFigure')
ans =
[]
```

### 9.3.2 Figure 对象用做 GUI

从复杂的应用程序到简单的对话框，都属于 GUI 的范畴。通过设置 Figure 对象的属性，可以修改图形窗口的很多特征，如下面的图形属性常用于创建 GUI：

- MenuBar　显示和隐藏图形窗口的菜单；
- Name　改变图形窗口的名称；
- HandleVisibility　控制图形句柄的可见性；
- ResizeFcn　创建回调，在用户改变图形窗口的大小时运行；
- Toolbar　控制图形窗口工具条的显示；
- UIContextMenu　指定一个上下文菜单；
- WindowButtonDownFcn,WindowButtonMotionFcn,WindowButtonUpFcn　定义回调，用户在图形窗口上单击、拖拉或释放鼠标时运行；
- WindowStyle　指定图形窗口类型。

### 9.3.3 Root 对象——Figure 对象的父对象

Figure 对象的父对象是 Root 对象。不能实例化 Root 对象，因为它的作用只是保存信息。它保存 MATLAB 状态、计算机系统和 MATLAB 默认设置等信息。

只有一个 Root 对象，其他对象都是它的子对象。Root 对象不用创建，启动 MATLAB 时就有了，但是，设置 Root 对象的属性值可以改变图形的显示。

## 9.4 核心图形对象

核心图形对象包括基本的绘图对象如直线、文本和多边形壳（Patch 对象）、曲面对象、图像对象和光照对象等。Axes 对象中包含有表示数据的对象，如直线、曲面和等值线组等。

表 9-1 列出了所有的核心图形对象和创建对应对象的函数。

表 9-1　核心图形对象

| 函　数 | 功　能 |
|---|---|
| axes | Axes 对象定义显示图形的坐标系统，它总是包含在 Figure 对象中 |
| image | 矩阵的二维表示，矩阵中的数值值映射为颜色。Image 对象还可以是 RGB 值的三维数组 |
| light | 坐标系内的镜面光源。Light 对象影响 Patch 对象和 Surface 对象的显示，但是它们本身不可见 |
| line | 连接定义直线的数据点，形成直线 |
| patch | 填充多边形集合。一个 Patch 对象可以由多个小面组成，每个小面可以单独着色，有刻面着色和插值着色两种着色方式 |
| rectangle | 矩形对象，可以绘制椭圆 |
| surface | 绘制矩阵数据的三维表面图 |
| text | 文本字符串 |

图 9-4 中显示了几种主要的核心图形对象。

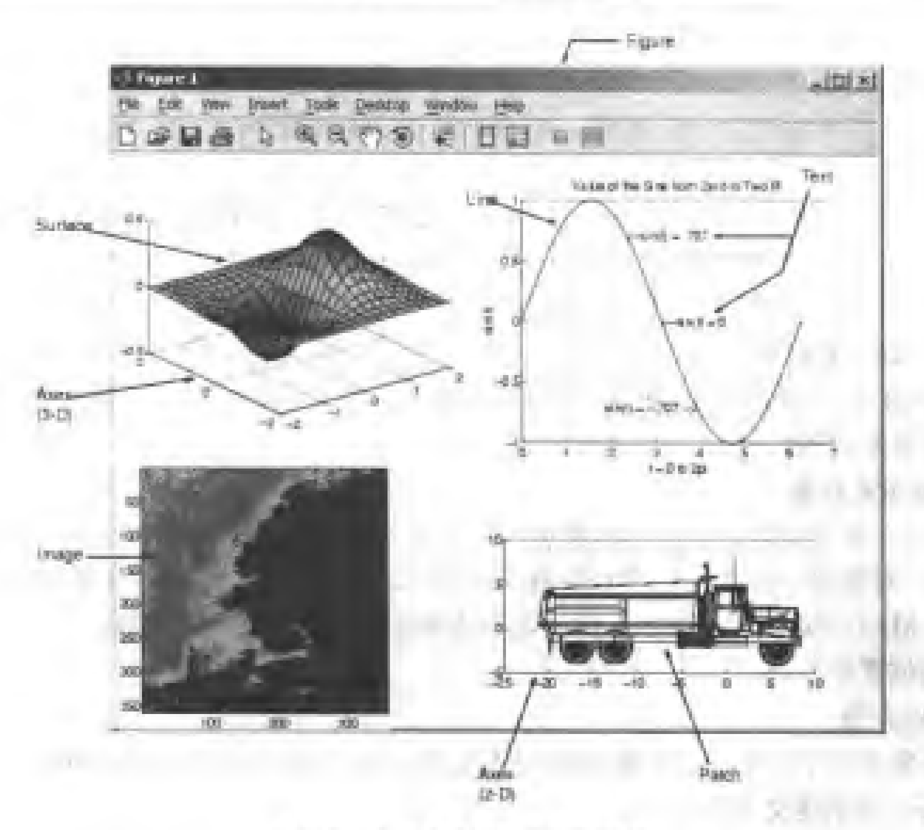



图 9-4 几种主要的核心图形对象

### 9.4.1 核心图形对象简介

下面对几种核心图形对象作一简单描述。在后续章节中，将进一步详细介绍这些对象。

#### 1. Axes 对象

Axes 对象在图形窗口中定义一个区域，并帮助这个区域内的子对象进行定向。Axes 对象是 Figure 对象的子对象，是 Image, Light, Line, Patch, Surface 和 Text 等对象的父对象。

如果当前图形窗口中没有 Axes 对象，则所有绘图函数绘图时都会创建一个该对象。如果在图形窗口中有多个 Axes 对象，则总有一个指定为当前坐标轴，当前绘制的图形对象都会显示在这里。

#### 2. Image 对象

Image 对象由一个数据矩阵组成，有时候可能还会有一个颜色映射矩阵。根据数据矩阵元素用像素颜色解释方式的不同，有 3 种基本的图像类型：索引图像、灰度图像和真彩色（RGB）图像。因为图像是严格二维的，索引只能在默认的二维视图中查看它们。

#### 3. Light 对象

Light 对象定义光源，这些光源会影响坐标系中所有 Patch 对象和 Surface 对象的显示效果。Light 对象是不可见的，但是可以像设置其他图形对象的属性那样设置光源类型、颜色、位置和其他属性。

#### 4. Line 对象

Line 对象是创建大部分二维图形和有些三维图形的基本图形元素。利用高级函数 plot, plot3 和 loglog 等可以创建 Line 对象。Axes 对象的坐标系统确定 Line 对象的位置和方向。

#### 5. Patch 对象

Patch 对象是经过填充的多边形。一个单独的 Patch 对象可以包含多种填充样式，可以用单色或渐变色进行填充。可以用 fill,fill3 和 contour3 函数创建 Patch 对象。父对象 Axes 确定 Patch 对象的位置和方向。

#### 6. Rectangle 对象

MATLAB 的 Rectangle 对象并不仅仅表示矩形，它可以表示矩形、椭圆及二者之间的过渡图形如圆角矩形等。

#### 7. Surface 对象

Surface 对象是矩阵数据的三维表示，它将每个矩阵元素作为 $x$-$y$ 平面上的高度进行显示。Surface 对象表示表面图，表面图由一系列四边形组成，这些四边形的顶点高度由矩阵数据指定。MATLAB 可以用单色、渐变色或连接各点的网格线绘制表面图。父对象 Axes 确定表面图的位置和方向。

#### 8. Text 对象

Text 对象表示字符串。父对象 Axes 定位这些文本。可以用高级函数 title, xlabel, ylabel, zlabel 和 gtext 等创建文本对象。

### 9.4.2 创建核心图形对象

可以用对象创建函数创建这些核心对象。所有对象创建函数都具有相似的格式，即

```
handle = function('propertyname',propertyvalue,...)
```

可以通过将属性名/属性值对作为变量进行传递来给所有对象属性指定值。该函数返回所创建的对象的句柄，可以用它查询和修改属性值。

下面的代码计算一个数学函数，并通过将属性值作为变量指定给 figure, axes 和 surface 函数来创建 3 个图形对象。MATLAB 对所有其他属性使用默认值。

```
[x,y] = meshgrid([-2:.4:2]);
Z = x.*exp(-x.^2-y.^2);
fh = figure('Position',[350 275 400 300],'Color','w');
ah = axes('Color',[.8 .8 .8],'XTick',[-2 -1 0 1 2],...
        'YTick',[-2 -1 0 1 2]);
sh = surface('XData',x,'YData',y,'ZData',Z,...
        'FaceColor',get(ah,'Color')+.1,...
        'EdgeColor','k','Marker','o',...
        'MarkerFaceColor',[.5 1 .85]);
```

生成的图形如图 9-5 所示。

三维视图的效果如图 9-6 所示。

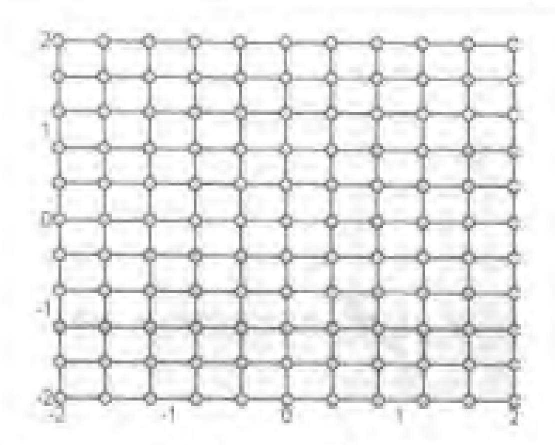


图 9-5 用句柄图形对象生成的图形窗口和图形

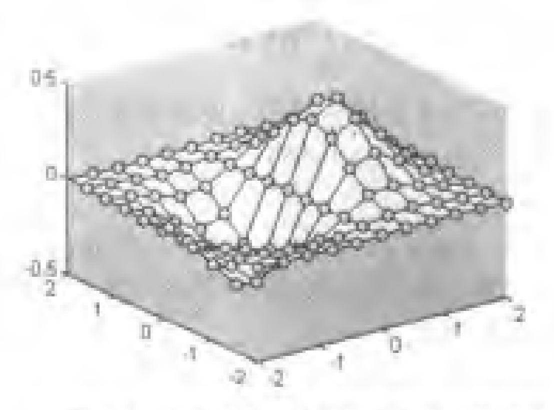


图 9-6 将图 9-5 中的图形调整到三维视图

可以用 view 命令改变视图，如

```
view(3)
```

### 9.4.3 父对象

默认时，所有语句都会在当前 Figure 对象和当前 Axes 对象中创建图形对象。但是，也可以在创建图形对象的过程中自己指定对象的父对象。例如，下面的语句

```
axes('Parent',figure_handle,...)
```

在 figure_handle 指定的图形窗口中创建一个坐标系。还可以通过重定义对象的 Parent 属性将一个对象从一个父对象移到另一个中，即

```
set(gca,'Parent',figure_handle)
```

### 9.4.4 高级函数和低级函数

MATLAB 高级图形函数如 plot 或 surf 等调用合适的对象创建函数来绘制图形对象。但是，根据 Axes 和 Figure 对象 NextPlot 属性的设置情况，高级函数可能还会清除坐标轴或创建新的图形窗口。

对比而言，对象创建函数只是简单地创建它们各自的图形对象并放到当前父对象中。它们不会理会 Axes 和 Figure 对象 NextPlot 属性的设置情况。

例如，如果调用 line 函数，即

```
line('XData',x,'YData',y,'ZData',z,'Color','r')
```

MATLAB 会用指定的数据值在当前坐标系统中绘制一条红色的直线段。如果没有坐标轴，MATLAB 会创建一个。如果没有图形窗口，MATLAB 也会创建一个。

如果第 2 次调用 line 函数，MATLAB 会将第 2 条直线段绘制到当前坐标系统中，第 1 条直线段仍然保留。这一特点与高级函数如 plot 不同，高级函数会删除图形对象然后重新设置坐标轴属性。可以用 hold 命令或通过改变 Axes 对象 NextPlot 属性的设置来改变高级函数的行为。

### 9.4.5　简化的调用语法

利用对象创建函数的简便形式可以更简单地进行调用，例如，

```
text(.5,.5,.5,'Hello')
```

等价于

```
text('Position',[.5 .5 .5],'String','Hello')
```

注意，使用对象创建函数的简便形式时得到的结果与使用一般形式时的比可能会有一些细微的差别。

## 9.5　绘图对象

用一系列高级绘图函数可以创建绘图对象。绘图对象的属性提供了获取核心图形对象重要属性的简便途径。

绘图对象的父对象可以是 Axes 或组对象（hggroup 或 ghtransform），如图 9-7 中所示。

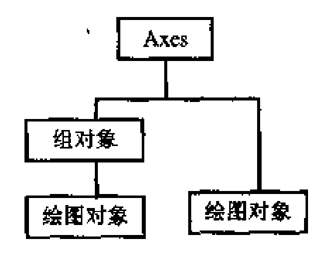



图 9-7　绘图对象的父对象

表 9-2 列出了绘图对象和使用它们的绘图函数。

表 9-2　绘图对象

| 对　象 | 功　能 |
|---|---|
| areaseries | 用于创建面积图 |
| barseries | 用于创建条形图 |
| contourgroup | 用于创建等值线图 |
| errorbarseries | 用于创建误差条图 |
| lineseries | 用于直线绘图函数(plot, plot3 等) |
| quivergroup | 用于创建二维和三维向量图 |
| scattergroup | 用于创建二维和三维散点图 |
| stairseries | 用于创建阶梯图 |
| stemseries | 用于创建二维和三维火柴杆图 |
| surfaceplot | 用于曲面和网格函数组 |

### 9.5.1　创建绘图对象

例如，下面的语句创建 peaks 函数的等值线图，然后设置等值线的线型和宽度。

```
[x,y,z] = peaks;
[c,h] = contour(x,y,z);
set(h,'LineWidth',3,'LineStyle',':')
```

生成图 9-8。

通过设置两个属性，可以用等值线绘图对象设置等值线图中的线型和线宽。察看包含在等值线图中的核心对象可以看到很多 Patch 对象，它们的边用于实现等值线。

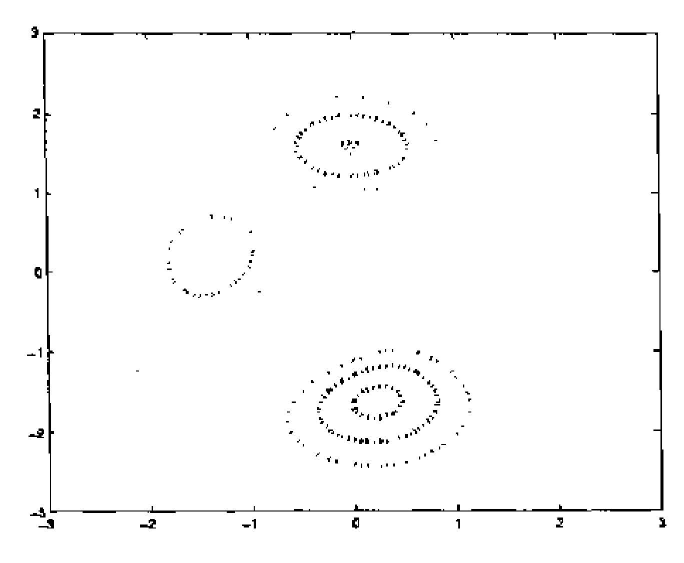


图 9-8　peaks 函数的等值线图

```
child_handles = get(h,'Children');
get(child_handles,'Type')
ans =
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
    'patch'
```

### 9.5.2　编程识别绘图对象

绘图对象都会把 ghgroup 作为 Type 属性的值返回。如果想通过编程识别图形对象，但是无法获得图形的句柄，可以给对象的 Tag 属性赋一个值。例如，下面的语句创建一幅有 5 个 barseries 对象的条形图，并为每个对象的 Tag 属性赋一个不同的值。

```
h = bar(rand(5));
set(h,{'Tag'},{'bar1','bar2','bar3','bar4','bar5'}')
```

注意，属性值的单元数组必须转置到合适的形式。

### 9.5.3　链接图形和变量

使用绘图对象可以链接 MATLAB 表达式和包含数据的属性。例如，lineseries 对象具有

与 XData,YData 和 ZData 相连的数据源属性 XDataSource,YDataSource 和 ZDataSource。

要正确使用数据源属性，必须按以下步骤进行：

- 将变量名指定给要与表达式链接的数据源属性；
- 为变量计算新值；
- 调用 refreshdata 函数更新绘图对象的数据。

使用 refreshdata 函数可以指定是否在基本工作空间中使用变量或指定调用 refreshdata 的函数的工作空间。

下面的例子演示如何使用这个技巧。

```
function datasource_ex
t = 0:pi/20:2*pi;
y = exp(sin(t));
h = plot(t,y,'YDataSource','y');
for k = 1:.1:10
    y = exp(sin(t.*k));
    refreshdata(h,'caller')
    drawnow; pause(.1)
end
```

如果将一个数据源属性变为包含不同维数据的变量，函数可能会生成一个警告信息，并且直到把所有数据源属性值变为合适值才绘图。

### 9.5.4　保存与 MATLAB 以前版本相兼容的图形

按照下面的步骤创建向后兼容的 FIG 文件：

- 确保任何用于创建图形窗口中内容的绘图函数调用时有“v6”选项；
- 使用带有“v6”选项的 hgsave 命令。

例如，

```
h = figure;
t = 0:pi/20:2*pi;
plot('v6',t,sin(t).*2)
hgsave(h,'myFigFile','-v6')
```

可以设置一个一般的 MATLAB 优先级来确保用“File”菜单下的“Save”选项保存图形时是向后兼容的。扩展“General”节点并选择“MAT Files”选项，单击“Ensure backward compatibility(-v6)”单选钮，如图 9-9 所示。注意，这个设置会影响所创建的所有 FIG 文件和 MAT 文件。




图 9-9　进行相关设置

## 9.6　Annotation 对象

通常，用户都是从工具条或“Insert”菜单中创建标注，但是，也可以用 annotation 函数进行创建。

Annotation 对象在隐藏坐标系中创建，该坐标系的宽度和高度与图形窗口的相同，这样，可以标准化坐标（左下角点的坐标为 0,0，右上角的坐标为 1,1）。在图形窗口中任意一处指定标注对象的位置。

Annotation 对象有 arrow, doublearrow, ellipse, line, rectangle, textarrow, textbox 等属性，设置这些属性的值，可以定义标注的内容和外观。修改绘图工具创建的标注对象的外观时，使用属性编辑器。

下面的例子显示如何创建矩形标注对象和用它亮显图形窗口中的两个子图。用坐标系属性 Position 和 TightInset 确定标注矩形的位置和大小。

（1）首先创建子图数组。

```
x = -2*pi:pi/12:2*pi;
y = x.^2;
subplot(2,2,1:2)
plot(x,y)
h1=subplot(223);
y = x.^4;
plot(x,y)
h2=subplot(224);
y = x.^5;
plot(x,y)
```

（2）用坐标系的 Position 属性和 TightInset 属性确定包围坐标系标记标签和标题的标注矩形的位置和大小。

```
t1 = get(h1,'TightInset');
p2 = get(h2,'Position');
t2 = get(h2,'TightInset');
x1 = p1(1)-t1(1); y1 = p1(2)-t1(2);
x2 = p2(1)-t2(1); y2 = p2(2)-t2(2);
w = x2-x1+t1(1)+p2(3)+t2(3); h = p2(4)+t2(2)+t2(4);
```

（3）创建包围下面两个子图的标注矩形，用透明的红色填充矩形。

```
annotation('rectangle',[x1,y1,w,h],...
'FaceAlpha',.2,'FaceColor','red','EdgeColor','red');
```

## 9.7　组对象

使用组对象可以把多个坐标系子对象作为一组进行处理。例如：可以使整个组可见或不可见；只单击一个组对象时，可选择该组所有对象；或用一个变换矩阵改变对象的位置等。

组对象有两个子对象：

- hggroup　创建一组对象或控制组的可见性或可选性时使用，用 hggroup 函数创建；
- hgtransform　试图变换一组对象时使用，变换包括旋转、平移和缩放等。

### 9.7.1　创建组对象

可以通过将坐标系子对象作为 hggroup 或 hgtransform 对象的子对象创建组。例如，

```
hb = bar(rand(5)); % 创建 5 个 barseries 对象
hg = hggroup;
set(hb,'Parent',hg) % 将 barseries 对象作为 hggroup 对象的子对象
set(hg,'Visible','off') % 使所有 barseries 对象不可见
```

组对象可以是任意个数坐标系子对象的父对象，包括其他组对象。

注意：许多绘图函数在绘制图形以前清除坐标系，清除坐标系还会删除坐标系中的所有 hggroup 或 hgtransform 对象。

### 9.7.2　变换对象

使用 hgtransform 对象的 Matrix 属性可以给所有 hgtransform 对象的子对象应用一个变换。典型的变换包括旋转、平移和缩放。用 4×4 的矩阵定义变换。

makehgtform 函数简化了进行旋转、平移和缩放时矩阵的创建工作。

旋转变换使对象绕 $x$, $y$ 或 $z$ 轴旋转，逆时针旋转时角度为正。如果旋转角度为 $\theta$，则下面的矩阵定义绕各轴的旋转。

$$\begin{bmatrix}1 & 0 & 0 & 0\\0 & \cos\theta_x & -\sin\theta_x & 0\\0 & \sin\theta_x & \cos\theta_x & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta_y & 0 & \sin\theta_y & 0\\0 & 1 & 0 & 0\\-\sin\theta_y & 0 & \cos\theta_y & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta_z & -\sin\theta_z & 0 & 0\\\sin\theta_z & \cos\theta_z & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

平移变换相对于当前位置移动对象，用距离 $t_x$, $t_y$ 和 $t_z$ 指定平移。下面的矩阵显示了这些元素在变换矩阵中的位置。

$$\begin{bmatrix}1 & 0 & 0 & t_x\\0 & 1 & 0 & t_y\\0 & 0 & 1 & t_z\\0 & 0 & 0 & 1\end{bmatrix}$$

缩放（比例）变换改变对象的大小，指定比例因子 $s_x$, $s_y$ 和 $s_z$ 创建下面的矩阵：

$$\begin{bmatrix}s_x & 0 & 0 & 0\\0 & s_y & 0 & 0\\0 & 0 & s_z & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

默认变换矩阵是单位矩阵，可以用 eye 函数创建它。下面是单位矩阵：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

变换是绝对意义上的，而不是相对于当前变换进行的。例如，如果应用一个变换，它在 $x$ 方向上把 hgtransform 对象平移了 5 个单位，然后用另一个变换把对象在 $y$ 方向上平移 4 个单位，则对象的最后位置是相对于原来位置在 $y$ 方向上平移了 4 个单位。如果希望作相对变换，必须把单个的变换矩阵聚合到一个单一的矩阵中。

通常，通过聚合单个矩阵，并把结果赋给 Matrix 属性来实现矩阵聚合更为有效。注意，矩阵相乘是不可逆的，所以矩阵的顺序会影响计算结果。例如，假设要进行一个先缩放，然后平移、旋转的操作，可以像下面这样作矩阵相乘的运算。

```
C = R*T*S % 从右向左计算
```

其中 $\boldsymbol{S}$ 是缩放矩阵，$\boldsymbol{T}$ 是平移矩阵，$\boldsymbol{R}$ 是旋转矩阵，而 $\boldsymbol{C}$ 是 3 种操作的组合。然后设置 hgtransform 对象的 Matrix 属性值为 $\boldsymbol{C}$。

```
set(hgtransform_handle,'Matrix',C)
```

注意，下面的语句行

```
set(hgtransform_handle,'Matrix',C)
set(hgtransform_handle,'Matrix',eye(4) )
```

与

```
C = eye(4)*R*T*S
set(hgtransform_handle,'Matrix',C)
```

是不等价的。

把单位矩阵聚合到其他矩阵中对于复合矩阵没有影响。

因为变换操作是在绝对意义上指定的，所以可以通过将当前变换设置为单位矩阵来实现一系列变换操作。例如，下面的语句

```
set(hgtransform_handle,'Matrix',eye(4))
```

把对象 hgtransform_handle 返回到它的变换前的方位。

因为旋转是绕原点进行的，常常需要平移 hgtransform 对象，这样，旋转的目标坐标暂时位于原点。采用旋转变换矩阵以后，把 hgtransform 对象移回原点。下面的例子演示如何做到这一点。假设要在曲面的中心绕 $y$ 轴旋转曲面。

（1）创建曲面和一个 hgtransform 对象，把 surface 对象作为 hgtransform 对象的子对象。例如，

```
h = surf(peaks(40)); view(-20,30)
t = hgtransform;
set(h,'Parent',t)
```

生成图 9-10 所示的曲面图。

（2）创建和设置一个绕 $y$ 轴旋转的矩阵，把曲面旋转 $-15°$

```
ry_angle = -15*pi/180;
Ry = makehgtform('yrotate',ry_angle);
set(t,'Matrix',Ry)
```

生成图 9-11 所示的曲面图。

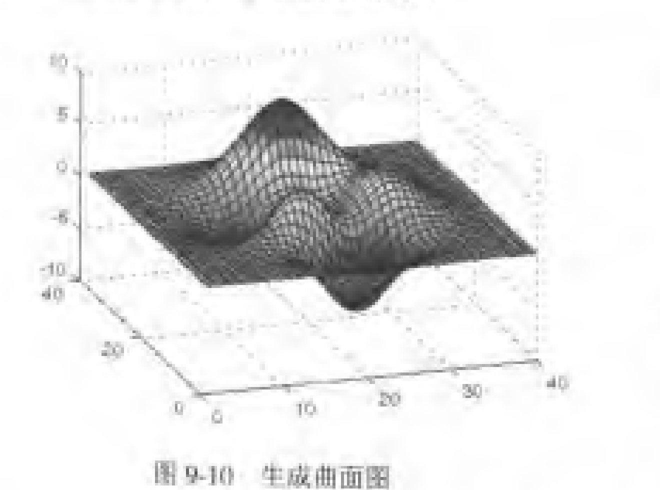


图 9-10　生成曲面图

图 9-11　绕 $y$ 轴旋转

（3）创建两个平移矩阵，一个在 $x$ 方向上把曲面平移-20 个单位，另一个往回移 20 个单位。按正确的顺序聚合两个平移矩阵和旋转矩阵并设置变换。

```
Tx1 = makehgtform('translate',[-20 0 0]);
Tx2 = makehgtform('translate',[20 0 0]);
set(t,'Matrix',Tx2*Ry*Tx1)
```

生成图 9-12。

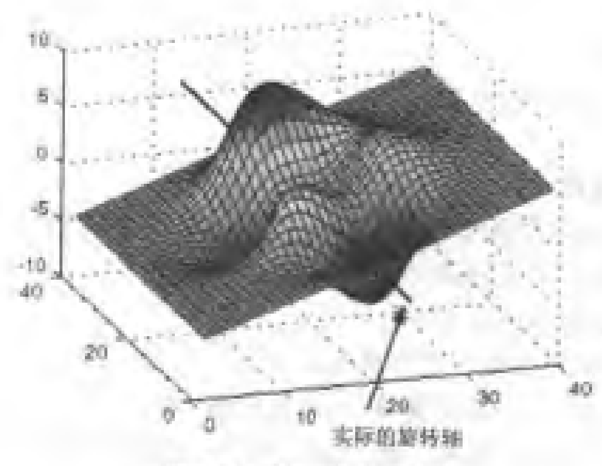



图 9-12　进一步设置变换

下面的例子创建一个 hgtransform 对象层次，然后先后旋转它们，用 6 个方形创建一个立方体。本例演示了如何把 hgtransform 对象作为其他 hgtransform 对象的父对象创建一个树形层次以及如何变换层次中的成员来影响子坐标系中的成员。图 9-13 显示了这个对象层次。

（1）创建图形窗口和视图。使用双缓冲来防止循环过程中出现闪动现象。把 Renderer 属性值设置为 zbuffer。

```
set(gcf,'DoubleBuffer','on');
set(gcf,'Renderer','zbuffer');
% 设置坐标系范围和视图
set(gca,'XLim',[0 4], 'YLim',[0 4], 'ZLim', [0 3]);
view(3); axis equal; grid on
```

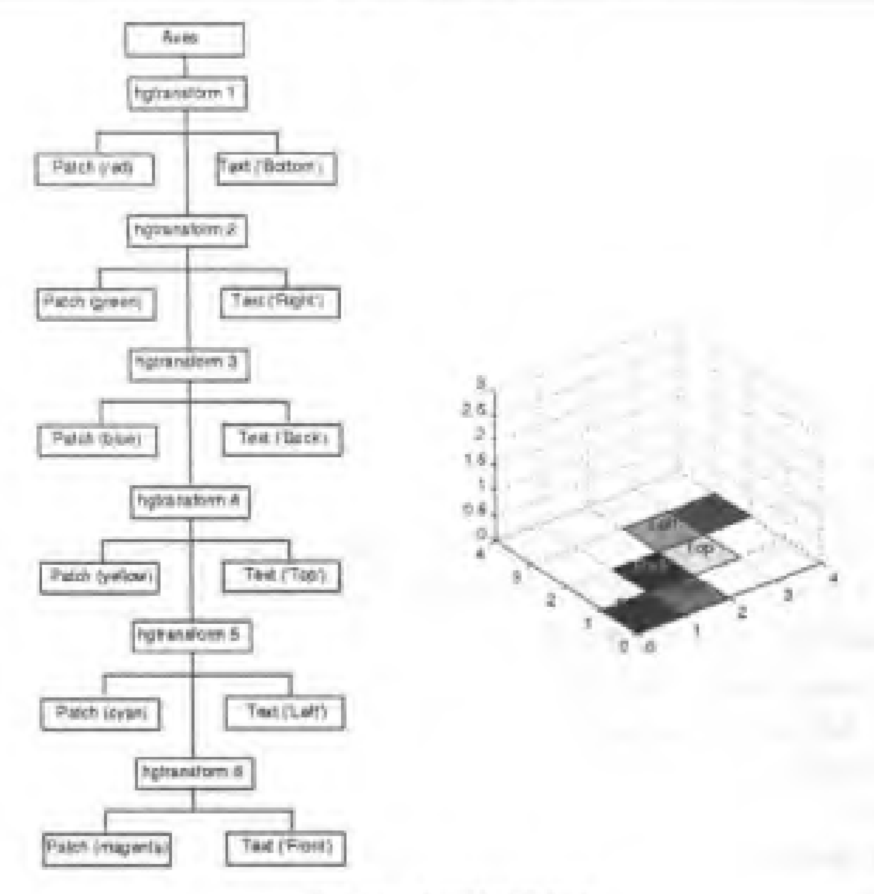



图 9-13　本例的对象层次

（2）定义 hgtransform 对象层次。

```
t(1) = hgtransform;
t(2) = hgtransform('parent',t(1));
t(3) = hgtransform('parent',t(2));
t(4) = hgtransform('parent',t(3));
t(5) = hgtransform('parent',t(4));
t(6) = hgtransform('parent',t(5));
```

（3）创建 Patch 和 Text 对象，并将每个 Patch-Text 对象对作为各 hgtransform 对象的父对象。

```
X = [0 0 1 1];
Y = [0 1 1 0];
Z = [0 0 0 0];
Xtext = .5;
Ytext = .5;
Ztext = .15;
p(1) = patch('FaceColor','red','Parent',t(1));
txt(1) = text('String','Bottom','Parent',t(1));
p(2) = patch('FaceColor','green','Parent',t(2));
txt(2) = text('String','Right','Parent',t(2));
```

```
p(3) = patch('FaceColor','blue','Parent',t(3));
txt(3) = text('String','Back','Color','white','Parent',t(3));
p(4) = patch('FaceColor','yellow','Parent',t(4));
txt(4) = text('String','Top','Parent',t(4));
p(5) = patch('FaceColor','cyan','Parent',t(5));
txt(5) = text('String','Left','Parent',t(5));
p(6) = patch('FaceColor','magenta','Parent',t(6));
txt(6) = text('String','Front','Parent',t(6));
set(p,'XData',X,'YData',Y,'ZData',Z)
set(txt,'Position',[Xtext Ytext Ztext],...
    'HorizontalAlignment','center',...
    'VerticalAlignment','middle')
```

（4）把方形（Patch 对象）平移到目标位置。注意，因为 hgtransform 对象 2 经过了平移，它的所有子对象也会平移。所以，每次平移都只需要把方形在 $x$ 或 $y$ 方向上移动一个单位，hgtransform 对象 1 还留在原来位置上。

```
Tx = makehgtform('translate',[1 0 0]);
Ty = makehgtform('translate',[0 1 0]);
set(t(2),'Matrix',Tx);
drawnow
set(t(3),'Matrix',Ty);
drawnow
set(t(4),'Matrix',Tx);
drawnow
set(t(5),'Matrix',Ty);
drawnow
set(t(6),'Matrix',Tx);
```

## 9.8　对象的属性

图形对象的属性控制它的外观和行为的很多方面。属性包括了诸如对象类型、父对象、子对象、可见性之类的一般信息，也包括该对象类型的独有信息。

例如，在任何给定的 Figure 对象上，可以知道最后按下的是哪个键、鼠标光标的位置或者最近选择的菜单的句柄等。

MATLAB 有层次地组织图形信息，并将该信息保存到属性中。例如，Root 对象的属性包含了当前图形窗口的句柄和鼠标光标的当前位置。

可以查询所有属性的当前值并指定大部分属性的值。注意，一个属性值只属于对象的一个实例，该对象设置一个属性值并不改变同类型其他对象的值。可以给属性设置默认值。默认值会影响随后创建的所有对象。

有些属性是所有对象都具备的，如表 9-3 所示。

表 9-3　所有对象都有的属性

| 属　性 | 包含的信息 |
|---|---|
| BusyAction | 控制 MATLAB 操作回调程序中断的方式 |
| ButtonDownFcn | 按键操作时执行的回调程序 |
| Children | 该对象所有子对象的句柄 |
| Clipping | 设置裁剪有效或无效的模式 |
| CreateFcn | 创建这种类型的对象时执行的回调程序 |
| DeleteFcn | 发出命令销毁对象时执行的回调程序 |
| HandleVisibility | 允许从命令行和回调程序内部控制对象句柄的有效性 |
| Interruptible | 确定一个回调程序是否能被随后调用的回调程序中断 |
| Parent | 对象的父对象 |
| Selected | 指示对象是否被选择 |
| SelectionHighlight | 指定对象是否可见地指示了选择状态 |
| Tag | 用户指定的对象标签 |
| Type | 对象类型 |
| UserData | 希望与对象连接的任何数据 |
| Visible | 确定对象是否可见 |

### 9.8.1　设置和查询属性值

set 和 get 函数指定和提取已经存在的图形对象属性值。设置对象属性值的基本语法格式为

```
set(object_handle,'PropertyName','NewPropertyValue')
```

用下面的语法查询指定对象属性的当前值：

```
returned_value = get(object_handle,'PropertyName');
```

**1. 设置属性值**

可以用 set 函数和创建函数返回的句柄改变对象的属性。例如，下面的语句将 $y$ 轴移到当前坐标系统的右侧

```
set(gca,'YAxisLocation','right')
```

如果句柄变量是一个向量，则 MATLAB 将指定值赋给所有可识别的对象。

可以用结构数组或单元数组指定属性名和属性值。如果是给很多对象设置相同的属性值，这种方法就很有用。例如，可以通过将 Axes 对象的属性值设置为结构来显示特定的图形。

```
view1.CameraViewAngleMode = 'manual';
view1.DataAspectRatio = [1 1 1];
view1.ProjectionType = 'Perspective';
```

将这些值设置给当前坐标轴，键入

```
set(gca,view1)
```

**2. 列出可能的属性值**

可以用 set 函数在没有实际指定一个新值的情况下显示属性的所有可能取值。例如，下面的语句获取所有可以指定为 Line 对象标记的值。

```
set(obj_handle,'Marker')
```

MATLAB 为 obj_handle 指定的对象类型的 Marker 属性返回一个取值列表，如下所示。其中，有括号的表示是默认值。

```
[ + | o | * | . | x | square | diamond | v | ^ | > | < | pentagram
| hexagram | {none} ]
```

要查看所有可以设置的属性及其属性值，只将对象句柄作为 set 函数的参数就可以了。即

```
set(object_handle)
```

例如，对于一个 Surface 对象，MATLAB 返回

```
CData
CDataScaling: [ {on} | off]
EdgeColor: [ none | {flat} | interp ] ColorSpec.
EraseMode: [ {normal} | background | xor | none ]
FaceColor: [ none | {flat} | interp | texturemap ] ColorSpec.
LineStyle: [ {-} | -- | : | -. | none ]
  ⋮
Visible: [ {on} | off ]
```

如果将 set 函数的输出指定给一个变量，则 MATLAB 将输出作为一个结构数组返回。例如，

```
a = set(gca);
```

a 中的字段名是对象的属性名，字段值是相关属性的可能值，例如，

```
a.GridLineStyle
ans =
    '-'
    '--'
    ':'
    '-.'
    'none'
```

返回坐标网格线的可能线型。注意，属性名没有大小写区分，但是 MATLAB 结构字段名有大小写区分。例如，

```
a.gridlinestyle
??? Reference to non-existent field 'gridlinestyle'.
```

返回一个错误。

**3．查询属性值**

用 get 函数查询一个属性或所有对象属性的当前值。例如，下面的代码检查当前 Axes 对象的 PlotBoxAspectRatio 属性的值。

```
get(gca,'PlotBoxAspectRatio')
ans =
  1     1     1
```

对于包含数据的属性，MATLAB 只列出维数。例如下面的 CurrentPoint 和 ColorOrder

属性。

```
AmbientLightColor = [1 1 1]
Box = off
CameraPosition = [0.5 0.5 2.23205]
CameraPositionMode = auto
CameraTarget = [0.5 0.5 0.5]
CameraTargetMode = auto
CameraUpVector = [0 1 0]
CameraUpVectorMode = auto
CameraViewAngle = [32.2042]
CameraViewAngleMode = auto
CLim: [0 1]
CLimMode: auto
Color: [0 0 0]
CurrentPoint: [ 2x3   double]
ColorOrder: [ 7x3   double]
  ⋮
Visible = on
```

对于这两个属性，可以通过单独查询它们来获取属性值。例如，对于 ColorOrder 属性，用 get 函数获取它的属性值。

```
get(gca,'ColorOrder')
ans =
         0         0    1.0000
         0    0.5000         0
    1.0000         0         0
         0    0.7500    0.7500
    0.7500         0    0.7500
    0.7500    0.7500         0
    0.2500    0.2500    0.2500
```

## 9.8.2 默认属性

### 1. 定义默认值

要指定默认值，需要创建一个以 Default 打头的字符串，后面跟对象类型和对象属性。例如，下面将 Line 对象的 LineWidth 属性的默认值设置为 1.4 磅。

```
set(gcf,'DefaultLineLineWidth',1.4)
```

同样，要指定图形的颜色，使用 DefaultFigureColor。注意，只有在 Root 对象级别上指定默认的图形颜色才是有意义的。

```
set(0,'DefaultFigureColor','b')
```

使用 get 函数确定当前设置了什么默认值。例如，

```
get(gcf,'default')
```

返回当前图形属性的所有默认值。

**2．删除默认值**

将属性值指定为'remove'，删除对应属性的默认值。例如，语句

```
set(0,'DefaultSurfaceEdgeColor','remove')
```

删除 Surface 对象 EdgeColor 属性的默认值。

### 9.8.3 示例——设置默认线型

**例 1** 本例创建一个图形窗口，背景色为白色，然后在 Root 对象级别上设置 Axes 对象的默认值。

```
whitebg('w')
set(0,'DefaultAxesColorOrder',[0 0 0],...
'DefaultAxesLineStyleOrder','-|--|:|-.')
Z = peaks; plot(1:49,Z(4:7,:))
```

不管什么时候调用 plot 函数都只用一种颜色绘制图形，因为 Axes 对象的 ColorOrder 属性只包含一种颜色。但是会循环使用 LineStyleOrder 属性中定义的线型。绘图效果如图 9-14 所示。

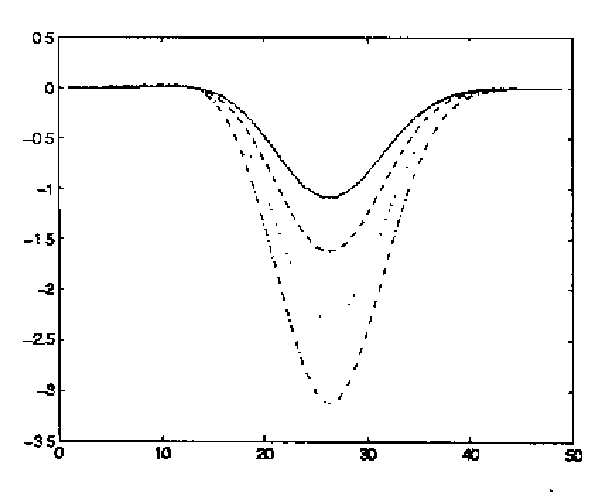



图 9-14 设置不同的线型

**例 2** 本例设置对象层次中一个以上级别上的默认值。下面这些语句在一个图形窗口中创建两个 Axes 对象，在 Figure 对象级别和 Axes 对象级别上设置默认值。

```
t = 0:pi/20:2*pi;
s = sin(t);
c = cos(t);
% 设置坐标系 Color 属性的默认值
figh = figure('Position',[30 100 800 350],...
              'DefaultAxesColor',[.8 .8 .8]);

axh1 = subplot(1,2,1); grid on
% 设置第一个坐标系中 Line 对象 LineStyle 属性的默认值
```

```
set(axh1,'DefaultLineLineStyle','-.')
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
HorizontalAlignment','right')

axh2 = subplot(1,2,2); grid on
% 设置第二个坐标系中 Text 对象 Rotation 属性的默认值
set(axh2,'DefaultTextRotation',90)
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
      HorizontalAlignment','right')
```

在不同的子图区域中使用 line 和 text 语句会获得不同的显示结果，它们反映了不同的默认设置。绘图效果如图 9-15 所示。

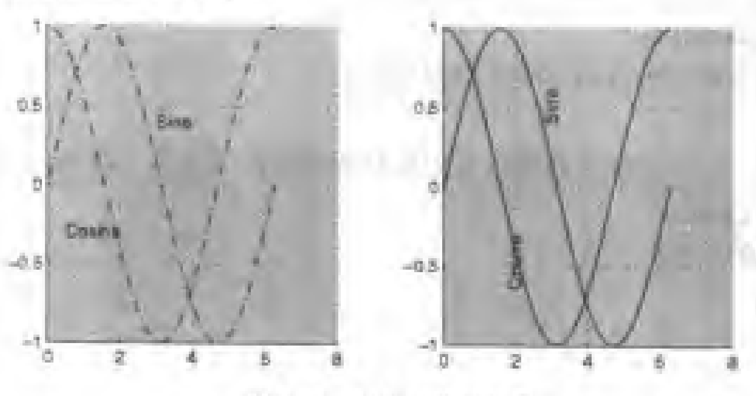



图 9-15　默认设置的效果

## 9.9　句柄操作

MATLAB 给每个新创建的图形对象指定一个句柄。所有对象创建函数都可以返回这个句柄。对句柄所作的任何操作，也作用在对应的图形对象上。所以，利用句柄，可以实现图形对象的查询、复制和删除等操作。

### 9.9.1　获取对象句柄

如果想获取对象的属性，就应该在创建该对象的时候将句柄指定给一个变量，以免以后用到它时再去查找。用 findobj 函数或列出其父对象的 Children 属性值也可以获取对象的句柄。

Root 对象的句柄总是 0。Figure 对象的句柄默认时是一个整型值，也可以是一个浮点值。可以用 IntegerHandle 属性控制图形接收相应的一类句柄。

### 9.9.2　当前图形、坐标轴和对象

与句柄图形有关的一个重要概念是“当前”。当前 Figure 对象是指定接收图形输出的窗口。类似地，当前 Axes 对象是包含目前创建的 Axes 对象子对象的坐标系。当前对象是最后创建的图形对象或最后用鼠标单击的对象。

可以用 Current 打头的属性来获取当前对象。例如，用 CurrentFigure 属性获取当前图形窗口，用 CurrentAxes 属性获取当前坐标轴，用 CurrentObject 属性获取当前对象等。

```
get(0,'CurrentFigure');
get(gcf,'CurrentAxes');
get(gcf,'CurrentObject');
```

下面的命令可看作是 get 语句的缩写形式。

gcf　返回 Root 对象 CurrentFigure 属性的值；
gca　返回当前 Figure 对象 CurrentAxes 属性的值；
gco　返回当前 Figure 对象 CurrentObject 属性的值。

可以将这些命令作为输入参数传递给需要对象句柄的函数。例如，可以单击一条直线段对象，然后用 gco 命令将句柄指定给 set 命令。

```
set(gco,'Marker','square')
```

或用下面的语句列出所有当前坐标轴属性的值。

```
get(gca)
```

用下面的语句，可以获取当前坐标系中所有图形对象的句柄（隐藏的句柄除外）。

```
h = get(gca,'Children');
```

然后确定对象的类型。

```
get(h,'type')
ans =
    text'
    patch'
    surface'
    line'
```

### 9.9.3　用属性值查找对象——findobj 函数

利用 findobj 函数，可以在对象层次中进行快速搜索并获得具有指定属性值的对象的句柄。如果没有指定初始对象，findobj 函数会从 Root 对象开始搜索，找出指定的所有属性名/属性值组合。

**例 1**　查找对象。

图 9-16 是一幅正弦函数图形，图中标注了函数的某些特定值。

假设将图中的标注文本“sin(t)=.707”从它的当前位置 [pi/4,sin(pi/4)] 移到点 [3*pi/4,sin(3*pi/4)]处，这两个点上函数具有相同的值（灰色显示）。要完成此项任务，需要确定该文本对象的句柄并改变它的 Position 属性。

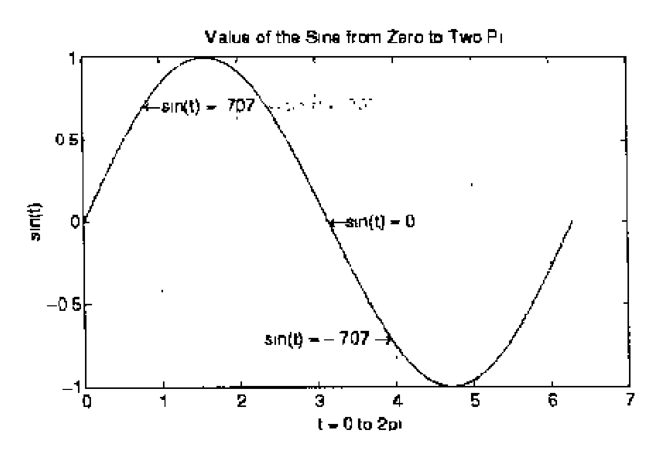



图 9-16　正弦函数图形

使用 findobj 函数，选择一个惟一标识该对象的属性值。本例使用了文本对象的 String 属性，即

```
text_handle = findobj('String','\leftarrowsin(t) = .707');
```

然后将文本移动到新位置：

```
set(text_handle,'Position',[3*pi/4,sin(3*pi/4),0])
```

通过在对象层次中指定初始点，findobj 函数还可以限定搜索范围。默认时从 Root 对象开始搜索，所以如果对象层次中有很多对象，这样会更快一些。在上面我们知道目标文本对象在当前坐标系中，所以可以键入

```
text_handle = findobj(gca,'String','\leftarrowsin(t) = .707');
```

**例 2**　使用逻辑操作符和正则表达式。

首先，创建图 9-17，希望进一步修改该图形的某些属性。

```
x = 0:30;
y = [1.5*cos(x);4*exp(-.1*x).*cos(x);exp(.05*x).*cos(x)]';
h = stem(x,y);
set(h(1),'Color','black',...
        'Marker','o',...
        'Tag','Decaying Exponential')
set(h(2),'Color','black',...
        'Marker','square',...
        'Tag','Growing Exponential')
set(h(3),'Color','black',...
        'Marker','*',...
        'Tag','Steady State')
```

生成图 9-17。

现在把极限设置为虚线，因为它是 Axes 对象的子对象。使用下面的语句，只获取这条虚线（基线）。

```
set(findobj(gca,'-depth',1,'Type','line'),'LineStyle','--')
```

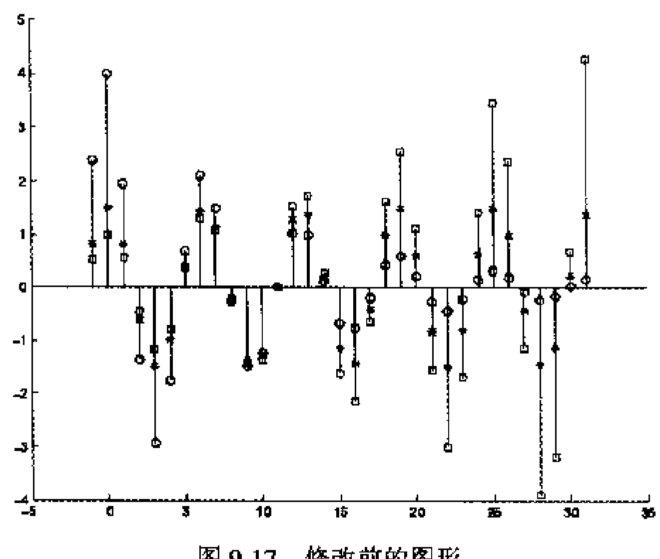


图 9-17　修改前的图形

通过将-depth 设置为 1，findobj 函数只搜索 Axes 对象和它的下一级子对象，就像从图中看到的，基线是惟一直接包含的 Line 对象。

下面的语句将所有 Tag 属性值没有设置为 Steady State 的火柴杆对象的 MarkerSize 属性值增加 2 磅

```
h = findobj('-regexp','Tag','[^Steady State]');
set(h,{'MarkerSize'},num2cell(cell2mat(get(h,'MarkerSize'))+2))
```

如果改变火柴杆的颜色而不改变标记，必须获取 3 个 stemseries 对象所包含的 Line 对象，不能只设置 stemseries 对象的 Color 属性，因为它即设置 Line 对象也设置标记的颜色。

下面的语句查找 Line 对象，要求 Marker 属性值为 none，LineStyle 属性值没有设置为“--”，它是基线。

```
h = findobj('type','line','Marker','none',...
            '-and','-not','LineStyle','--');
set(h,'Color','red')
```

如图 9-18 所示。

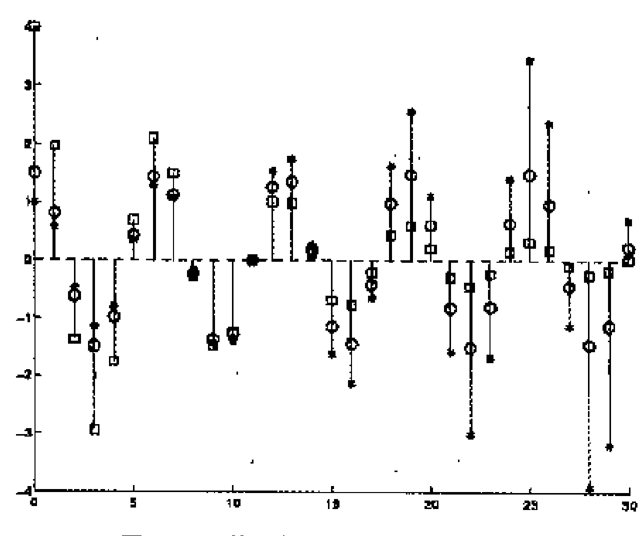


图 9-18　修改图形属性以后的结果

### 9.9.4　复制对象

可以用 copyobj 函数将对象从一个父对象复制到另一个父对象。新对象与原对象的惟一区别在于它的 Parent 属性和它的句柄。可以将多个对象复制到一个新的父对象中，或者将一个对象复制到多个新的父对象中。复制一个具有子对象的对象时，MATLAB 会一起复制所有子对象。

假设根据数据绘图并且在每幅图中标注 $x$ 和 $y$ 坐标由 $5\pi/4$ 和 $\sin(5\pi/4)$确定的点。利用 text 函数可以指定标签的位置。

```
text('String','\{5\pi\div4, sin(5\pi\div4)\}\rightarrow',...
     Position',[5*pi/4,sin(5*pi/4),0],...
     HorizontalAlignment','right')
```

此语句中，text 函数在图中用 $T_EX$ 命令标注{5π/4, sin(5π/4)}，在图中绘制向右箭头和数学标记，用 Position 属性指定标注的位置，通过将 HorizontalAlignment 属性改变为 right，将数据点放在文本字符串的右侧。标注效果如图 9-19 中所示。

在另一幅图中用同一字符串标注相同的点时，用 copyobj 函数复制文本。因为最后一个语句没有保存文本对象的句柄，所以用 findobj 函数和 String 属性查找它，即

```
text_handle = findobj('String',...
     \{5\pi\div4,sin(5\pi\div4)\}\rightarrow');
```

创建下一幅图以后，通过从第一幅图中复制来实现标签的添加。

```
copyobj(text_handle,gca).
```

其结果如图 9-20 中所示。

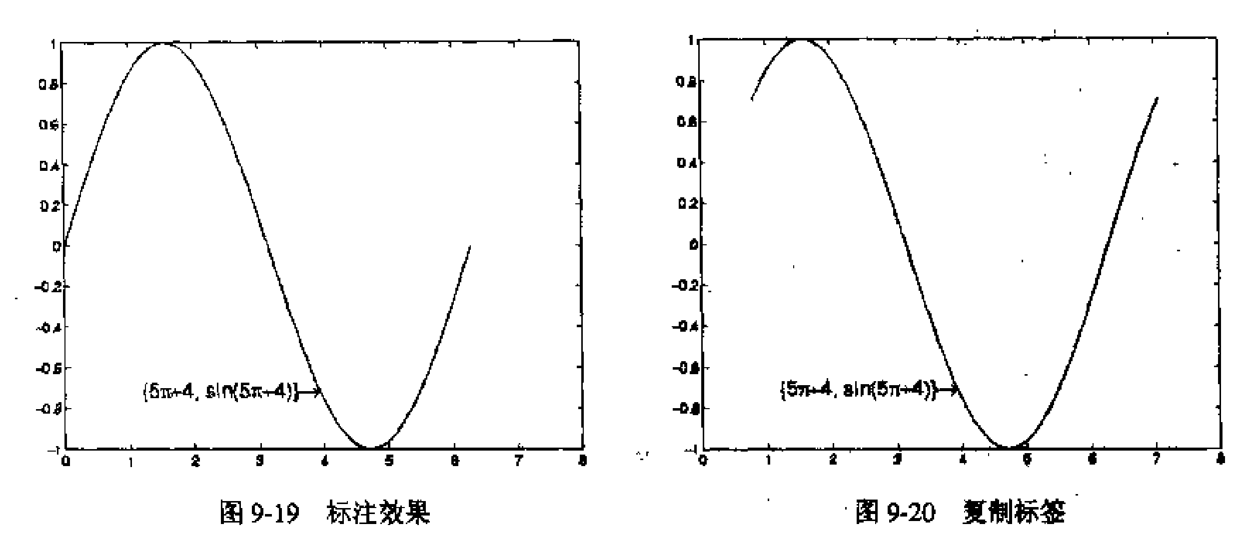



图 9-19　标注效果　　　　图 9-20　复制标签

这个例子利用了文本对象在坐标系数据空间中定义位置的特点。所以，文本的 Position 属性不需要随不同图形改变。

### 9.9.5　删除对象

用 delete 命令删除图形对象。例如，可以用下面的命令删除当前 Axes 对象。

```
delete(gca)
```

可以用 findobj 函数获取要删除的对象的句柄。例如，要找到图 9-21 中点线的句柄，用 findobj 函数查找 LineStyle 属性为':'的对象。

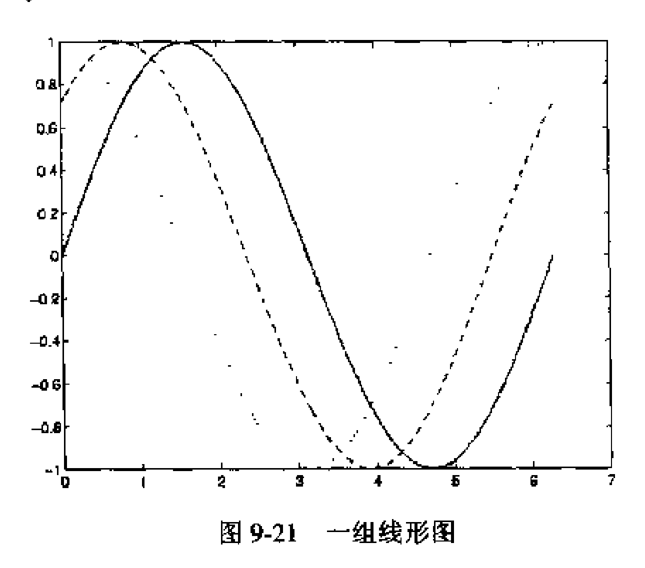


图 9-21　一组线形图

```
line_handle = findobj('LineStyle',':');
```

然后将返回的句柄传递给 delete 命令，删除该对象。

```
delete(line_handle)
```

可以将上面两行语句组合成下面的一行。

```
delete(findobj('LineStyle',':'))
```

## 9.10　句柄图形的视图控制

本节讨论控制 MATLAB 图形显示位置和方式的方法。包括：

- 指定图形输出的目标区域；
- 设置图形窗口和坐标系；
- 防止图形窗口和坐标系成为图形输出的目标区域；
- 获取不让输出图形的图形窗口和坐标系的句柄。

### 9.10.1　指定图形输出的目标区域

默认时，创建图形对象的 MATLAB 函数会将图形显示在当前图形窗口和当前坐标系中。可以通过指定 Parent 属性将输出指定给另一个父对象。例如，

```
plot(1:10,'Parent',axes_handle)
```

其中，axes_handle 是要输出图形的坐标系的句柄。使用 uicontrol 和 uimenu 函数下面的语法可以将父对象指定为第一个参数进行传递。

```
uicontrol(Figure_handle,...)
uimenu(parent_menu_handle,...)
```

### 9.10.2 设置图形窗口和坐标系

默认时，生成输出图形的命令会将图形对象显示在当前图形窗口中，图形窗口的属性不会清除和重置。但是，如果图形对象是 Axes 对象的子对象，则 MATLAB 会清除坐标系并在显示对象以前将大部分坐标系属性设置为它们的默认值。

可以通过设置 Figure 对象和 Axes 对象的 NextPlot 属性进行改变。

**1．用 NextPlot 属性控制输出**

MATLAB 高级图形函数通过检查 NextPlot 属性的值来确定是否在绘图以前添加、清除或清除和重置图形窗口和坐标系的属性。低级函数不检查 NextPlot 属性的值，它们只是简单地将新生成的图形对象添加到当前图形窗口和坐标系中。

低级函数主要用在 M 文件中。开发一个基于 MATLAB 的应用程序时，控制 MATLAB 的绘图行为对于创建有预见性的程序来说是很重要的。

表 9-4 概括了 NextPlot 属性所有可能的取值。

**表 9-4　NextPlot 属性值及对应的图形窗口和坐标系**

| NextPlot 属性取值 | 图形窗口 | 坐标系 |
| --- | --- | --- |
| add | 在不清除和重置当前图形窗口设置的情况下添加新的图形对象 | 在不清除或重置当前坐标系的情况下添加一个新的图形对象 |
| replacechildren | 删除所有图形对象，但是不重置图形对象的属性。等价于 clf 函数 | 删除所有子对象，但不重置坐标系属性。等价于 cla |
| replace | 删除所有子对象并将图形窗口的属性重置为默认值。等价于 clf reset | 删除所有子对象并将坐标系属性重置为默认值。等价于 cla reset |

注意，除了 Position 属性和 Units 属性以外，重置操作会将各属性的值设置为默认值。

hold 命令提供了设置 NextPlot 属性值的便捷方式。例如，语句

```
hold on
```

将图形窗口和坐标系的 NextPlot 属性值都设置为 add。语句

```
hold off
```

将坐标系的 NextPlot 属性值设置为 replace。

**2．用 newplot 函数设置目标输出区域**

newplot 函数检查 NextPlot 属性的值并基于这些值进行合适的操作。应该将 newplot 放在调用对象创建函数的 M 文件的开头。当 M 文件调用 newplot 函数时，可能产生下面一些结果：

（1）newplot 函数检查当前 Figure 对象的 NextPlot 属性：

- 如果当前没有 Figure 对象，则 newplot 函数创建一个并使它成为当前 Figure 对象。
- 如果 NextPlot 属性的值为 add，则 newplot 函数使该 Figure 对象成为当前 Figure 对象。
- 如果 NextPlot 属性的值为 replacechildren，则 newplot 函数删除该 Figure 对象的子对象并使该 Figure 对象成为当前 Figure 对象。
- 如果 NextPlot 属性的值为 replace，则 newplot 函数删除 Figure 对象的子对象，将

Figure 对象的属性设置为默认值，并使该 Figure 对象成为当前 Figure 对象。

（2）newplot 函数检查当前 Axes 对象的 NextPlot 属性：

- 如果当前没有 Axes 对象，则 newplot 函数创建一个并使它成为当前 Axes 对象。
- 如果 NextPlot 属性的值为 add，则 newplot 函数使该 Axes 对象成为当前 Axes 对象。
- 如果 NextPlot 属性的值为 replacechildren，则 newplot 函数删除该 Axes 对象的子对象并使该 Axes 对象成为当前 Axes 对象。
- 如果 NextPlot 属性的值为 replace，则 newplot 函数删除 Axes 对象的子对象，将 Axes 对象的属性设置为默认值，并使该 Axes 对象成为当前 Axes 对象。

### 3. MATLAB 的默认行为

默认情况下，Figure 对象的 NextPlot 属性值为 add，Axes 对象的 NextPlot 属性值为 replace。调用 newplot 函数时，它完成下面两个操作：

（1）检查当前 Figure 对象的 NextPlot 属性值并确定 MATLAB 是否能够在当前 Figure 对象上绘图。

（2）检查当前 Axes 对象的 NextPlot 属性值，从坐标系中删除所有图形对象，将 Axes 对象的所有属性设置为默认值，并返回当前 Axes 对象的句柄。

### 4. 示例——使用 newplot 函数

下面的例子演示 newplot 函数的使用方法和结果。首先创建一个与内部函数 plot 功能相近的函数，不同的是，它自动用不同线型来表示不同的多义线，而 plot 函数用不同的颜色来表示。

```
function my_plot(x,y)
cax = newplot;
LSO = ['- ';'--';': ';'-.'];
set(cax,'FontName','Times','FontAngle','italic')
set(get(cax,'Parent'),'MenuBar','none') %
line_handles = line(x,y,'Color','b');
style = 1;
for i = 1:length(line_handles)
    if style > length(LSO), style = 1;end
    set(line_handles(i),'LineStyle',LSO(style,:))
style = style + 1;
end
grid on
```

函数 my_plot 用 line 函数绘制数据。line 函数不检查 Figure 或 Axes 对象 NextPlot 属性的值。但是，因为 my_plot 函数调用 newplot 函数，所以它的工作方式与高级 plot 函数的相同，即每次调用它时 my_plot 函数清除 Axes 对象的属性值并将它们设置为默认值。

my_plot 函数用 newplot 函数返回的句柄获取当前 Figure 对象和 Axes 对象。下面的例子设置坐标轴的字体属性并使图形窗口的菜单条不可用。注意图形窗口的句柄是如何通过 Axes 对象的 Parent 属性获得的。

图 9-22 显示了 my_plot 函数绘制的图形，它通过下面的语句创建：

```
my_plot(1:10,peaks(10))
```

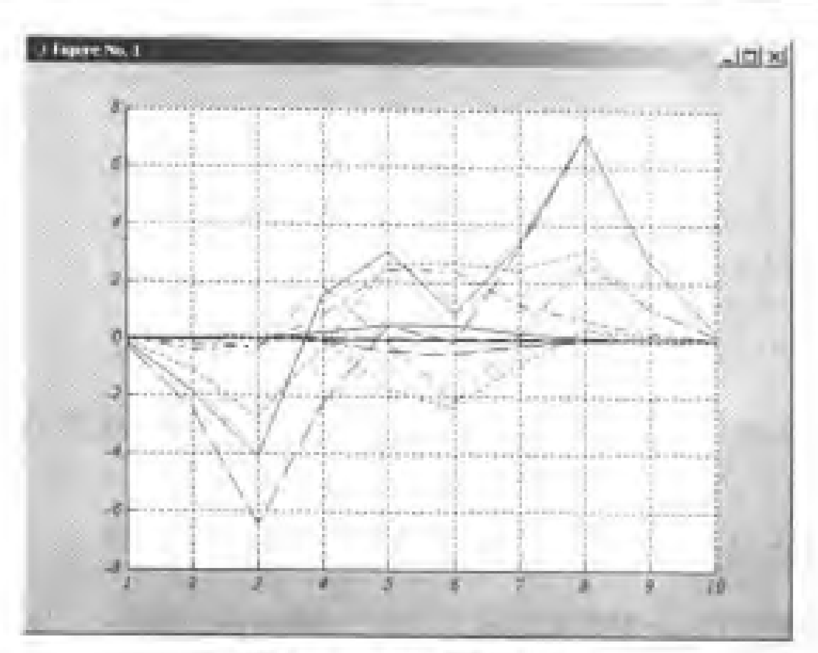



图 9-22　my_plot 函数绘制的图形

注意，图 9-22 内图形窗口中的菜单和工具条都没有显示，而且以后生成的图形窗口也没有菜单和工具条的显示。

### 9.10.3　测试持续绘图（Hold）状态

有时可能出现需要改变坐标系的外观来适应新图形对象的情况。例如，如果希望前面创建的 M 文件 my_plot 接受三维数据，则输入数据有 z 坐标时将视图设置为三维是有意义的。

但是，为了与 MATLAB 高级函数的行为保持一致，在改变父 Axes 或 Figure 对象的属性值以前测试一下 hold 属性是否处于 on 状态是很好的。hold 属性值为 on 时，Axes 和 Figure 对象的 NextPlot 属性都设置为 add。

下面的 M 文件 my_plot3 接受三维数据并用 ishold 函数检查 hold 属性的状态，以确定是否应该改变视图。

```
function my_plot3(x,y,z)
cax = newplot;
hold_state = ishold;
LSO = ['- ';'--';': ';'-.'];
if nargin == 2
    hlines = line(x,y,'Color','k');
    if ~hold_state
        view(2)
    end
elseif nargin == 3
    hlines = line(x,y,z,'Color','k');
    if ~hold_state
```

```
            view(3)
        end
    end
    ls = 1;
    for hindex = 1:length(hlines)
        if ls > length(LSO),ls = 1;end
        set(hlines(hindex),'LineStyle',LSO(ls,:))
        ls = ls + 1;
    end
```

如果调用 my_plot3 函数时 hold 属性处于 on 状态，则不改变视图；否则根据输入变量是 2 个还是 3 个，将视图设置为二维或三维。

### 9.10.4　防止 Figure 和 Axes 对象成为绘图目标区域

有时要防止某些 Figure 对象或 Axes 对象成为绘图目标区域对象，比如一个包含了多个实现用户界面的 Uicontrol 对象的 Figure 对象。可以通过将它的句柄从对 newplot 和其他函数可见的列表中删除来防止 MATLAB 将图形绘制到这些对象上。有两个属性可以控制句柄隐藏：HandleVisibility 属性和 ShowHiddenHandles 属性。

**1．HandleVisibility 属性**

HandleVisibility 属性是所有对象都有的属性。它控制句柄在 3 个不同范围内的可见性。属性值可以是：

- on　对象句柄对于任何在 MATLAB 命令行或 M 文件中执行的函数都是可用的。这是默认设置。
- callback　对于所有在命令行上执行的函数，对象句柄隐藏。但回调函数执行的过程中，句柄对所有函数是可见的。该设置使回调函数可以利用 MATLAB 句柄获取函数，并确保用户进行命令行操作时不会无意中干扰受保护的对象。
- off　对象的句柄对于运行在命令行和回调程序中的函数都是隐藏的。

如果一个图形用户界面（GUI）接受文本字符串形式的用户输入，然后在回调程序中进行处理。此时，一个诸如'close all'的字符串会导致该 GUI 关闭。为了防止出现这种情况，可以将关键对象的 HandleVisibility 属性暂时设置为 off。例如，

```
user_input = get(editbox_handle,'String');
set(gui_handles,'HandleVisibility','off')
eval(user_input)
set(gui_handles,'HandleVisibility','commandline')
```

**2．ShowHiddenHandles 属性**

ShowHiddenHandles 属性可以打开和取消句柄的可见性控制。默认时，ShowHiddenHandles 属性值为 off，表示 MATLAB 按照 HandleVisibility 属性设置进行操作。设置为 on 时，命令行和回调函数中所有句柄都可见。当希望获得一定时间内存在的所有图形对象，包括一般情况下不可见的坐标系文本标签句柄时，ShowHiddenHandles 属性的这一特点很有用。使用 hidden 选项，close 函数可以获得不可见的图形窗口。例如，

```
close('hidden')
```

会关闭屏幕上最顶层的窗口，即使该窗口是受保护的窗口也会关闭。组合 all 和 hidden 选项时，即

```
close('all', 'hidden')
```

关闭所有图形窗口。

### 9.10.5　关闭请求函数

关闭图形窗口时，MATLAB 执行一个回调函数，该函数由 Figure 对象的 CloseRequestFcn 属性定义。

使用关闭请求函数可以防止或延迟图形窗口的关闭，在需要用对话框要求用户确认操作，在关闭以前保存数据和防止在 GUI 内无意中删除命令行时很有用。

ColoseRequestFcn 属性的默认回调函数是一个名为 closereq 的 M 文件。它包含下面的语句：

```
shh=get(0,'ShowHiddenHandles');
set(0,'ShowHiddenHandles','on');
delete(get(0,'CurrentFigure'));
set(0,'ShowHiddenHandles',shh);
```

## 9.11　把句柄保存到 M 文件

图形 M 文件常常用句柄获取属性值并将图形输出指定给特定的目标区域。MATLAB 提供了一些工具函数将句柄返回给关键对象。但是，在 M 文件中，这些工具未必是获取句柄的最好方法，因为

- 在 MATLAB 中查询对象句柄或其他信息没有将句柄保存到变量然后引用该变量有效；
- 与用户交互的缘故，运行 M 文件时当前 Figure 对象、Axes 对象或其他对象可能会发生改变。

在 M 文件的开头保存 MATLAB 状态的相关信息是一个好的做法。例如，编写 M 文件时可以这样开始：

```
cax = newplot;
cfig = get(cax,'Parent');
hold_state = ishold;
```

可以避免每次需要时都查询该信息。

如果想暂时更改 M 文件中的 hold 状态，应该暂时保存 NextPlot 属性的当前值，使得以后可以重新设置它。例如，

```
ax_nextplot = lower(get(cax,'NextPlot'));
fig_nextplot = lower(get(cfig,'NextPlot'));
   ⋮
set(cax,'NextPlot',ax_nextplot)
set(cfig,'NextPlot',fig_nextplot)
```

## 9.12　可包含其他对象的对象

有些图形对象可以包含其他对象。考虑一幅图形，图中数据用类似 Line 的对象表示。正常情况下，Line 对象的父对象是 Axes 对象（即 Line 对象 Parent 属性的句柄设置为包含它的 Axes 对象的句柄）。Figure 对象通常是 Axes 对象的父对象。典型的对象图如图 9-23 所示。

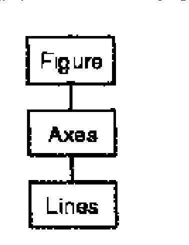



图 9-23　Figure 对象与其他对象的包含关系

当图形更复杂并用多个对象表示数据时，将这些对象组合在一起可以把它们作为一个组进行操作。

下面介绍如何使用两个容器对象。它们把图中的坐标系子对象和用户界面到一个图形窗口中。

Figure 对象可以直接包含坐标系和用户界面对象，或者可以将这些对象作为 UIPanels 的子对象。设计 GUI 时 uipanel 对象很有用，因为利用它们可以定义图形窗口中的子域。

MATLAB 定义 uipanel 对象的子对象的 Position 属性时是相对于 uipanel 的位置解释的。如果移动 uipanel 对象，它的子对象会自动随着移动。

uipanel 对象也可以包含其他 uipanel 对象，以及 Axes,uicontrol 和 uibuttongroup 对象。

下面的例子用 3 个 uipanel 对象作为 GUI 组件的容器。然后，这 3 个 uipanel 对象都作为 Figure 对象的子对象，如图 9-24 中所示。

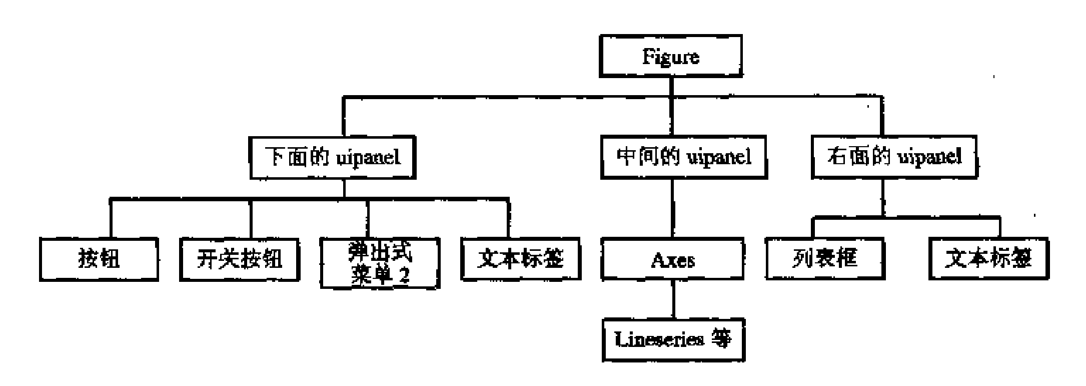



图 9-24　本例中各对象间的关系

图 9-25 是 GUI 中的绘图效果。

使用图形用户界面可以从列表框中选择工作空间变量或从弹出式菜单中选择一种绘图类型。可以通过单击“Hold”按钮在已经存在的图中添加图形，或者单击“Create Plot”按钮初始化图形。

### 1．创建 uipanel 对象

下面的代码显示了图形图形窗口下面面板的定义，将 Units 属性设置为 characters 可以确保 GUI 在不同计算机系统上正确改变大小。Position 属性指定位置，Units 属性指定大小。

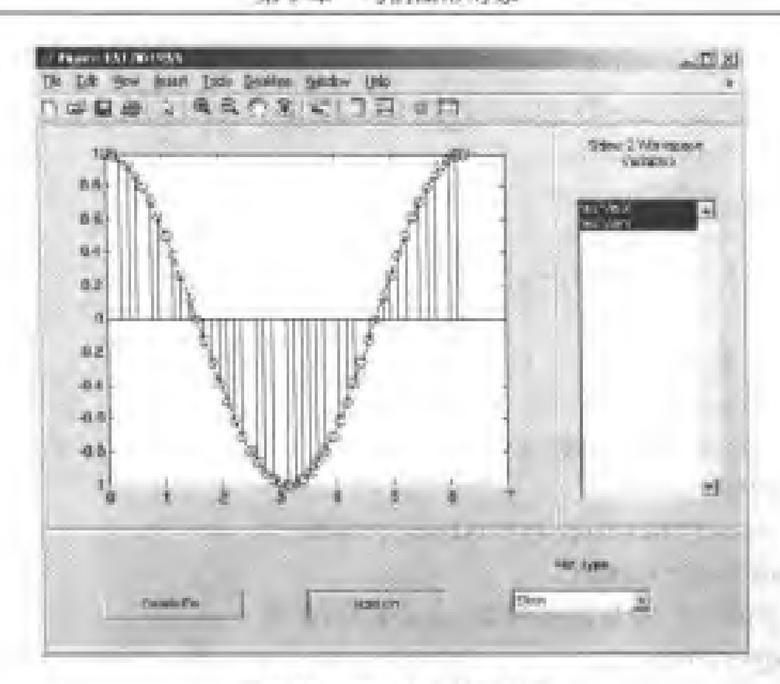


图 9-25　GUI 中的绘图效果

```
%创建图形窗口
f = figure('Units','characters',...
        'Position',[30 30 120 35],...
        'Color',panelColor,...
        'HandleVisibility','callback',...
        'IntegerHandle','off',...
        'Renderer','painters',...
        'ResizeFcn',@figResize);
%创建底部的 uipanel 对象
botPanel = uipanel('BorderType','etchedin',...
        'BackgroundColor',panelColor,...
        'Units','characters',...
        'Position',[1/20 1/20 119.9 8],...
        'Parent',f,...
        'ResizeFcn',@botPanelResize);
```

### 2. 编制图形窗口缩放函数

改变图形窗口的大小时，MATLAB 会调用相应的函数（用对象的 ResizeFcn 属性指定）。在下面的例子中，它计算每个控件新的大小。因为图形窗口缩放函数会改变控件的大小，每次窗口缩放函数完成运行时 MATLAB 都会自动调用每个控件的缩放函数。然后控件缩放函数调整它们所包含的组成元素的大小和位置。

下面的代码显示图形窗口、底部面板和右侧面板的缩放函数。当每个函数被调用时，相对于原来的情况按比例设置对象的大小和位置。

```
% 图形窗口缩放函数
function figResize(src,evt)
    fpos = get(f,'Position');
```

```
        set(botPanel,'Position',...
            [1/20 1/20 fpos(3)-.1 fpos(4)*8/35])
        set(rightPanel,'Position',...
          [fpos(3)*85/120 fpos(4)*8/35 fpos(3)*35/120 fpos(4)*27/35])
        set(centerPanel,'Position',...
            [1/20 fpos(4)*8/35 fpos(3)*85/120 fpos(4)*27/35]);
    end
    % 底部面板缩放函数
    function botPanelResize(src,evt)
        bpos = get(botPanel,'Position');
        set(plotButton,'Position',...
            [bpos(3)*10/120 bpos(4)*2/8 bpos(3)*24/120 2])
        set(holdToggle,'Position',...
            [bpos(3)*45/120 bpos(4)*2/8 bpos(3)*24/120 2])
        set(popUp,'Position',...
            [bpos(3)*80/120 bpos(4)*2/8 bpos(3)*24/120 2])
        set(popUpLabel,'Position',...
            [bpos(3)*80/120 bpos(4)*4/8 bpos(3)*24/120 2])
    end
    %右侧面板缩放函数
    function rightPanelResize(src,evt)
        rpos = get(rightPanel,'Position');
        set(listBox,'Position',...
            [rpos(3)*4/32 rpos(4)*2/27 rpos(3)*24/32 rpos(4)*20/27]);
        set(listBoxLabel,'Position',...
            [rpos(3)*4/32 rpos(4)*24/27 rpos(3)*24/32 rpos(4)*2/27]);
    end
```

注意，中间的面板不需要缩放函数，因为坐标系会自动改变大小来适应容器。

**3．对坐标系中包含的对象进行分组——hgtransform 对象**

MATLAB 提供了两个对象来实现对所有 Axes 子对象的分组，它们是 hggroup 和 hgtransform 对象。希望把对象作为一个组进行引用时将所有对象作为一个 hggroup 对象的子对象。例如，选择或控制组中所有成员的可见性时可作此类处理。使用 ghtransform 对象可以把对象作为一个组进行变换。

## 9.13 句柄图形对象的回调

### 9.13.1 图形对象的回调属性

Callback 属性是一个函数，它在特定事件发生在图形对象上时执行。通过设置合适的对象属性可以指定回调。下面介绍可以定义回调的事件。

所有图形对象都有 3 个属性，利用这 3 个属性，可以定义回调函数，它们是：

- ButtonDownFcn　当鼠标光标位于对象上方或者在对象周围 5 像素以内，单击鼠标左键时执行；
- CreateFcn　创建对象时，当所有属性都设置好以后执行；
- DeleteFcn　在删除对象以前执行。

1．Uicontrol, Uimenu 和 Uicontextmenu 对象的回调

Uicontrol、Uimenu 和 Uicontextmenu 对象都有一个 Callback 属性，可以用该函数定义激活这些对象时要执行的函数。

2．Figure 对象的回调

Figure 对象有其他属性执行回调，与用户的操作有关。只有 CloseRequestFcn 属性具有默认定义的回调。这些属性包括：

- CloseRequestFcn　当发出关闭图形窗口的请求时执行；
- KeyPressFcn　当鼠标光标位于图形窗口内，单击时执行；
- ResizeFcn　用户改变图形窗口大小时执行；
- WindowButtonDownFcn　当鼠标光标位于背景、不可用的控件或坐标系背景上方且单击时执行；
- WindowButtonMotionFcn　当鼠标光标在图形窗口内移动时执行；
- WindowButtonUpFcn　当用户在图形窗口中已经按下鼠标，然后释放鼠标时执行。

### 9.13.2　函数句柄回调

句柄图形对象有很多属性可以用于定义回调函数。当指定的事件发生时，执行对应的回调函数。可以将回调属性的值指定为：

- 表示 MATLAB 命令或 M 文件名称的字符串；
- 字符串单元数组；
- 函数句柄或包含函数句柄和其他变量的单元数组。

本节介绍如何为句柄图形对象定义函数句柄回调。

1．函数句柄的语法格式

句柄图形中，用作函数句柄回调的函数必须至少定义两个输入变量，一个是生成回调的对象的句柄，另一个是事件数据结构。不管什么时候执行回调，MATLAB 都会传递这两个变量。例如，对于一个 M 文件中的下面一段语句：

```
function myGui
figure('WindowButtonDownFcn',@myCallback)
      ⋮
function myCallback(obj,eventdata)
```

第 1 个语句创建一个 Figure 对象并将一个函数句柄指定给它的 WindowButtonFcn 属性。这个函数句柄指向子函数 myCallback。定义这个子函数时必须指定两个必要的输入变量。

将其他变量添加到函数定义行，可以定义回调函数接受其他变量，例如，

```
function myCallback(obj,eventdata,arg1,arg2)
```

使用其他变量时，必须将属性值设置为单元数组。例如，

```
figure('WindowButtonDownFcn',{@myCallback,arg1,arg2})
```

## 9.14　Figure 对象

Figure 图形对象表示 MATLAB 的绘图窗口。使用 Figure 对象的属性，可以控制绘图窗口的很多方面，例如它们在屏幕上的大小和位置，图形对象的着色，以及打印图形的比例等。

下面主要介绍一些用 Figure 对象的属性实现的特性，并提供了如何使用这些特性的实例。

### 9.14.1　在面板上锚定图形窗口

可以通过单击 按钮将图形窗口锚定在 MATLAB 面板上。锚定以后，图形窗口放置在图形窗口组容器中，该容器也可以锚定和取消锚定。

在图形窗口组容器中，可以选择不同的图形窗口摆放方式。图 9-26 显示了如何选择不同的图形窗口摆放方式。一旦锚定以后，图形窗口就亮显工具条和菜单条。

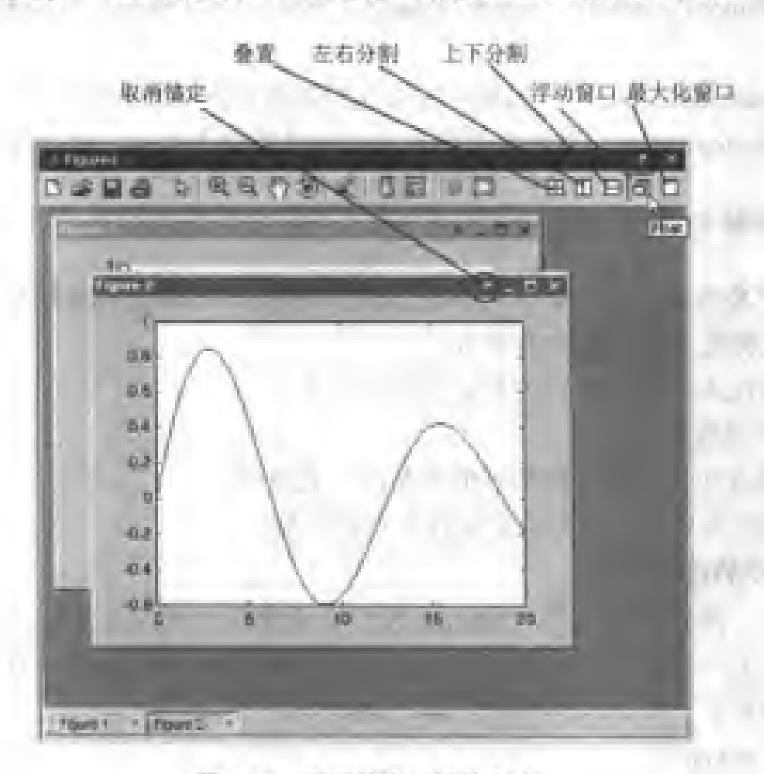



图 9-26　不同的窗口摆放方式

### 9.14.2　与窗口锚定有关的属性

有两种与图形窗口锚定有关的属性：DockControls 和 WindowStyle。

- DockControls 属性　该属性控制锚定图形窗口的控件显示。将 DockControls 属性的值设置为 off，将 按钮从菜单条上删除，并且"Desktop"菜单中的对应选项不可用。
- WindowStyle 属性　将 WindowStyle 属性设置为 docked 时，MATLAB 将面板上的图形窗口锚定在一个图形窗口组容器中。如果 WindowStyle 属性的值设置为 docked，则

MATLAB 自动将 DockControls 属性的值设置为 on，并且不能将 DockControls 属性的值设置为 off，不能设置 Position 属性。

如果不想让用户锚定图形窗口，应该如下设置图形窗口的属性：

- 将 DockControls 属性的值设置为 off；
- 将 WindowStyle 属性的值设置为 normal 或 modal；
- 将 HandleVisibility 属性的值设置为 off 或 callback。

### 9.14.3 确定图形窗口的位置和大小

Figure 对象的 Position 属性控制图形窗口在屏幕上的大小和位置。启动时，MATLAB 会根据计算机屏幕的大小定义 Position 属性的默认值。默认时创建的图形窗口大小约为屏幕大小的四分之一。

MATLAB 将图形窗口的 Position 属性值定义为向量，即

```
[left bottom width height]
```

其中，left 和 bottom 定义窗口左下角像素的坐标，相对于屏幕左下角指定；width 和 height 定义窗口内部的大小，如图 9-27 所示。

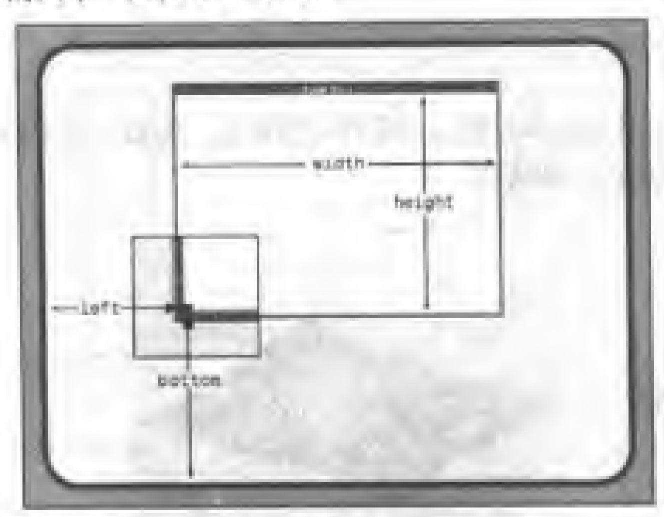



图 9-27　定义图形窗口的位置

确定图形窗口的位置时，MATLAB 不把窗口的边界计算在内，Position 属性由图形窗口的内部活动区域定义。

Figure 对象的 Units 属性确定值在屏幕上的度量单位。可以选用的度量单位有 inches,centimeters,normalized,points,pixels 和 characters 等，如下设置：

```
set(gcf,'Units')
[ inches | centimeters | normalized | points | {pixels} | characters]
```

其中，pixels 是默认选项，即用像素作为默认度量单位。

不管使用什么单位，了解用这些单位度量时屏幕的大小很重要。可以从 Root 对象的 ScreenSize 属性获取此信息。例如，

```
get(0,'ScreenSize')
ans =
     1 1 1152 900
```

本例中，屏幕大小为 1152×900 像素。MATLAB 按照 Root 对象 Units 属性确定的单位返回 ScreenSize 属性的值，例如，

```
set(0,'Units','normalized')
```

正规化 ScreenSize 属性返回的值：

```
get(0,'ScreenSize')
ans =
    0 0 1 1
```

## 9.15 坐标系属性

MATLAB 用图形表示数据，但不管是哪种图形，都需要有一个作为参考的图框，这个参考图框其实就是 Axes 对象定义的坐标系统。Axes 对象确定图形的方位和大小，生成数据视图。

Axes 对象总是包含在一个 Figure 对象中，它本身包含有组成图形的图形对象。Axes 对象的属性控制 MATLAB 图形显示的很多方面。下面介绍通过 Axes 对象属性实现的一些特性，并提供了如何使用这些特性的实例。

### 9.15.1 标签和外观属性

MATLAB 提供了很多标注和控制坐标系外观的属性。例如，图 9-28 所示的曲面图中显示了一些标注要素和对应的控制属性。

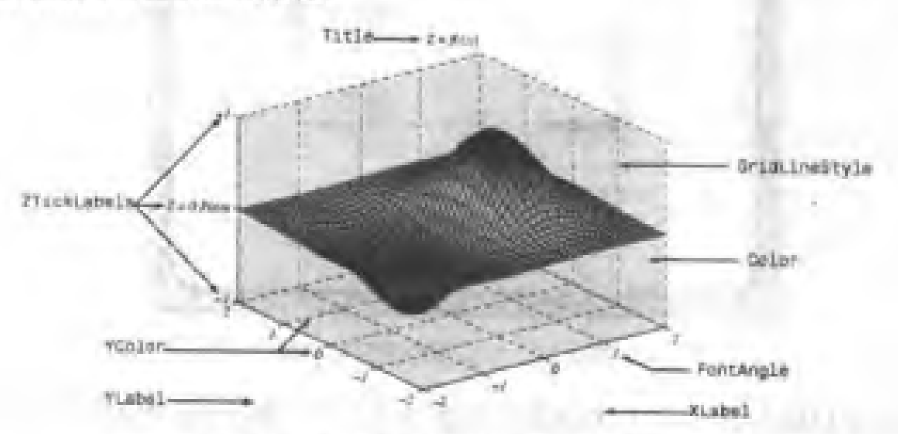



图 9-28 曲面图中的标注要素和控制属性

创建坐标系，需要给相关属性赋值，例如，

```
h = axes('Color',[.9 .9 .9],...
    'GridLineStyle','--',...
    'ZTickLabels','-1|Z = 0 Plane|+1',...
    'FontName','times',...
    'FontAngle','italic',...
    'FontSize',14,...
    'XColor',[0 0 .7],...
    'YColor',[0 0 .7],...
    'ZColor',[0 0 .7]);
```

坐标轴标签是文本对象，其句柄在命令行中通常不可见。可以用 Xlabel, Ylabel, Zlabel 和 Title 等函数创建坐标系标签。但是，这些函数只影响当前坐标系。如果通过引用坐标系句柄来标注当前坐标系以外的坐标系，则必须根据对应的坐标系属性获取文本对象的句柄。

```
get(axes_handle,'XLabel')
```

返回用作 $x$ 轴标签的文本对象。从坐标系获取文本句柄在 M 文件和不能确定目标对象是否当前坐标系的基于 MATLAB 的应用程序中很有用。下面的语句为上面的坐标系定义 $x$ 轴、$y$ 轴的标签和标题。

```
set(get(axes_handle,'XLabel'),'String','Values of X')
set(get(axes_handle,'YLabel'),'String','Values of Y')
set(get(axes_handle,'Title'),'String','\fontname{times}\itZ =
f(x,y)')
```

因为标签是文本，所以必须给 String 属性指定一个值，该属性的初值为空。MATLAB 会覆盖很多其他文本属性来控制这些标签的位置和方位。但是，可以设置 Color, FontAngle, FontName, FontSize, FontWeight 和 String 等属性。

注意，坐标系对象和文本对象都有指定字体的属性。下面的语句设置 FontName, FontAngle 和 FontSize 等属性。如果想给标签和标题使用相同的字体，则定义它们的 String 属性时指定相同的属性值。

```
set(get(h,'XLabel'),'String','Values of X',...
            'FontName','times',...
            'FontAngle','italic',...
            'FontSize',14)
```

### 9.15.2 坐标系的位置和大小

Axes 对象的 Position 属性控制图形窗口中坐标系的大小和位置。默认的坐标系具有与默认图形窗口相同的纵横比并填充除了周边以外的大部分图形窗口。

MATLAB 把坐标系的 Position 属性值设置为向量，即

```
[left bottom width height]
```

left 和 bottom 定义图中的点，该点对应于坐标系矩形的左下角，width 和 height 指定坐标系矩形的大小。在工作平面内察看坐标系时 $x$ 轴水平，$y$ 轴垂直。从这个角度讲，图形的包围矩形与坐标系矩形一致。二维和三维情况下坐标系位置和大小的度量如图 9-29 和图 9-30 所示。

默认时，MATLAB 绘制一个包围矩形来填充坐标系矩形并忽略它的形状，但是，使用坐标系属性可以控制图形包围盒的形状和大小。

Axes 对象的 Units 属性确定 Position 属性的度量单位，该属性的可能值为

```
set(gca,'Units')
[ inches | centimeters | {normalized} | points | pixels ]
```

其中，normalized 是默认值。不管图形有多大，正规化单位将图形左下角映射为(0,0)，右上角为(1,1)。不管什么时候改变图形窗口的大小，正规化单位会使坐标系自动改变大小。

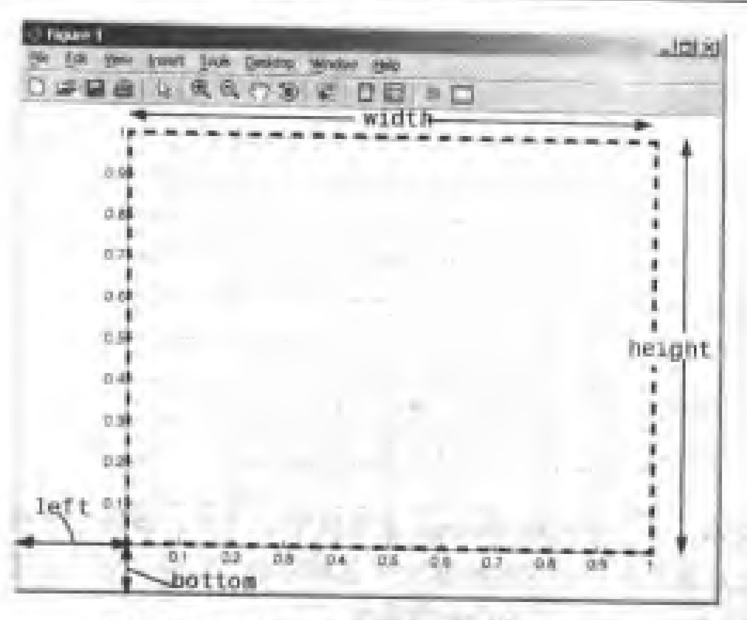



图 9-29　二维条件下坐标系的位置和大小度量

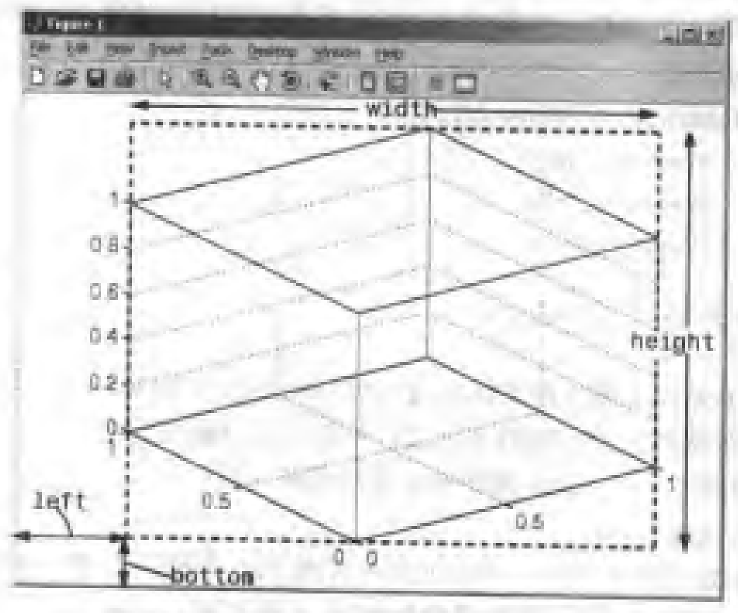



图 9-30　三维条件下坐标系的位置和大小度量

可以用下面的属性控制改变坐标系大小的行为：

- OuterPosition　坐标系的边缘包括坐标系标签、标题和空白，对于只有一个坐标系的图形窗口，它是图形窗口的内边缘。
- Position　除了标记、标签、标题和坐标轴标签等内容以外的坐标系的边界。
- ActivePositionProperty　指定改变包含坐标系的图形窗口的大小时是否使用 OuterPosition 或 Position 属性改变大小。
- TightInset　在坐标系周边留出空白，以便留出标签、标题和坐标系的标签的位置。
- Units　将该属性设置为 normalized，使坐标系自动改变大小。

图 9-31 显示了 OuterPosition,TightInset+Position 和 Position 属性定义的区域。

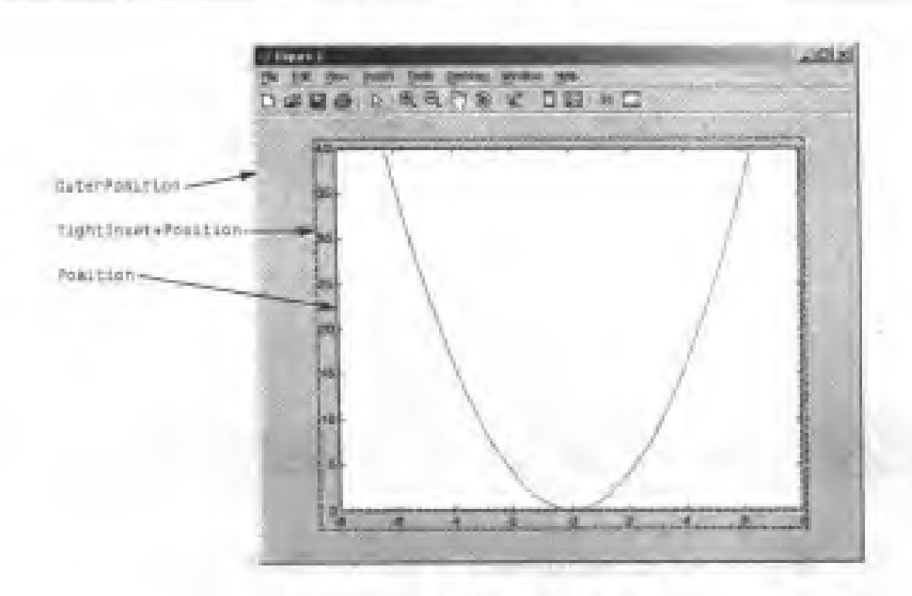



图 9-31 几种属性定义的区域

添加坐标轴标签和标题时，TightInset 属性会把这些标签和标题也包括在内，如图 9-32 中所示。

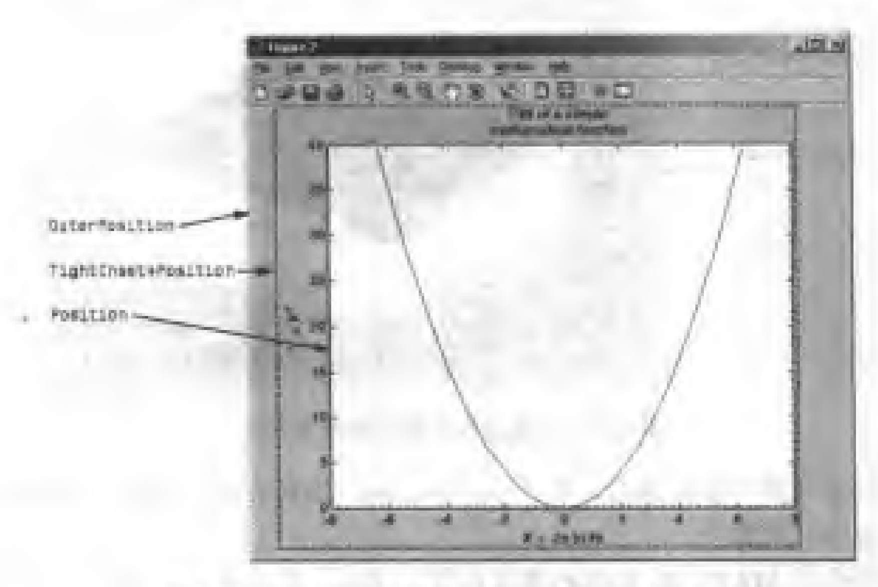
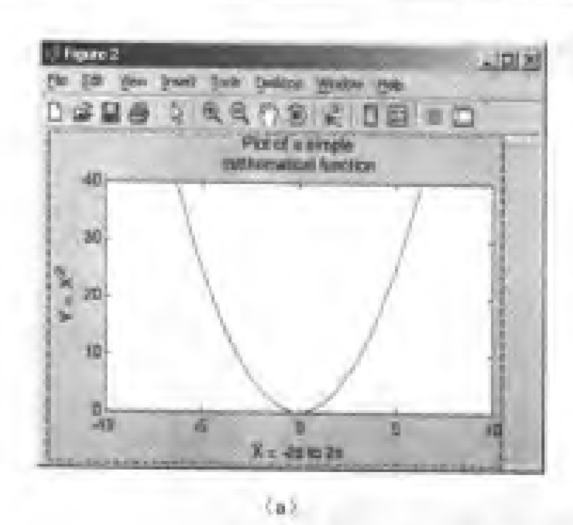



图 9-32 添加坐标轴标签和标题时属性定义的区域

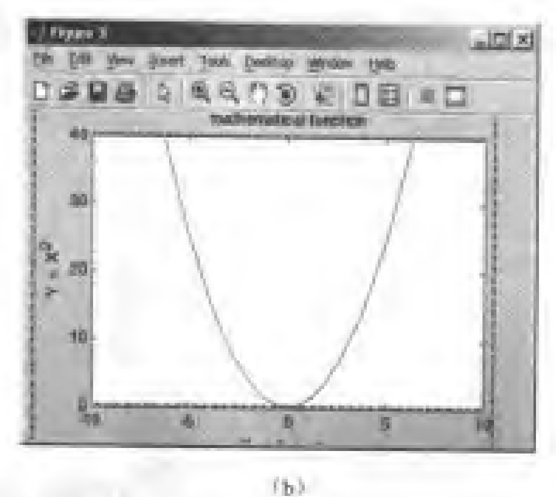
现在 TightInset 和 Position 属性定义的矩形大小包括所有的图形文本。Position 和 OuterPosition 属性的值保持不变。

改变图形窗口的大小时，MATLAB 会保持 TightInset 和 Position 定义的区域。比较图 9-33 中的图（a）和图（b），它们已经被改变为具有相同的图形窗口大小。其中，图（a）对应于 ActivePositionProperty 属性值为 OuterPosition 的情况；图（b）对应于 ActivePositionProperty 属性值为 Position 的情况。

图 9-34 显示了将这些属性应用于三维图形时的效果。



（a）



（b）

图 9-33　ActivePositionProperty 属性取不同值时坐标系的大小

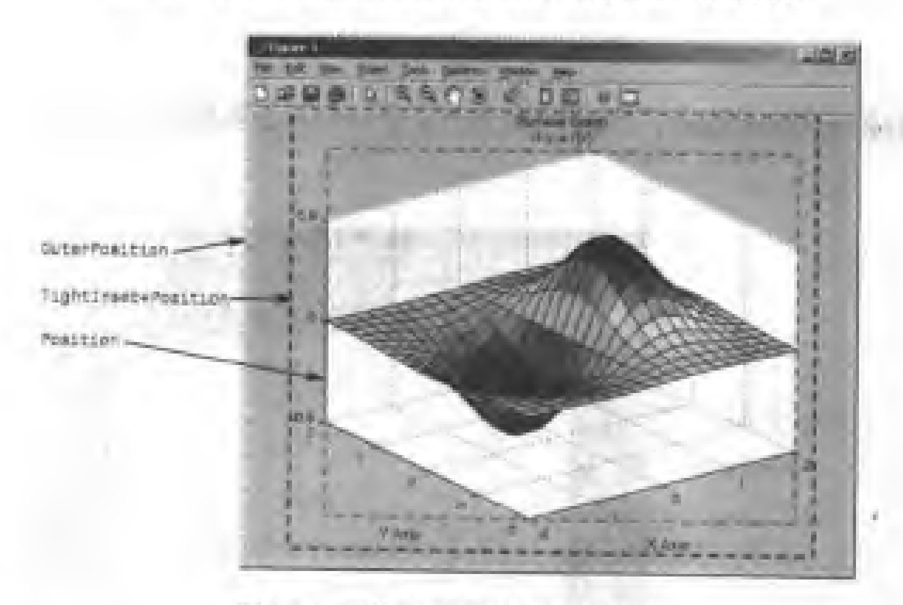



图 9-34　三维情况下属性的设置效果

当图形窗口中有多个坐标系时，把 OuterPosition 用作 ActivePositionProperty 是防止标题和标签被覆盖的有效途径。

图 9-35 演示了 MATLAB 如何改变坐标系的大小，以便在下面的两个坐标系中包含多行文本。

默认的三维视图方位角等于-37.5°，仰角等于 30°。

### 9.15.3　在同一图形窗口中显示多个坐标系

使用 subplot 函数可以在一个图形窗口中显示多个坐标系。

subplot 函数可以在图形窗口中输出多个等间隔的图形。但是，重叠的坐标系可以创建一些有用的效果。

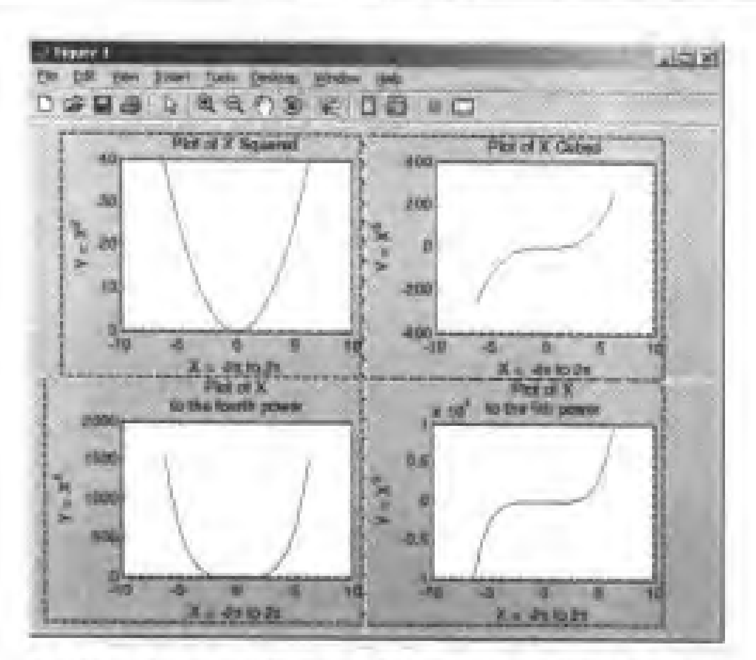


图 9-35　通过改变坐标系大小来实现在坐标系中包含多行文本

MATLAB 总是在坐标系中显示文本对象。如果想创建图形并在图形周边提供对图形的描述，必须另外创建一个坐标系来放置文本。如果创建一个与图形窗口大小相同的坐标系，然后创建一个更小的坐标系绘图，就可以独立于图形在任何位置显示文本。

例如，下面的代码定义两个坐标系。

```
h = axes('Position',[0 0 1 1],'Visible','off');
axes('Position',[.25 .1 .7 .8])
```

因为坐标系单位已经正规化到图形窗口，把 Position 属性指定为[0 0 1 1]会创建一个压缩整个窗口的坐标系。

现在在当前坐标系中绘制一些数据的图形，最后创建的坐标系是当前坐标系，所以 MATLAB 直接在那里绘图。

```
t = 0:900;
plot(t,0.25*exp(-0.005*t))
```

定义文本并在整个图形坐标系中显示它。

```
str(1) = {'Plot of the function:'};
str(2) = {' y = A{\ite}^{-\alpha{\itt}}'};
str(3) = {'With the values:'};
str(3) = {' A = 0.25'};
str(4) = {' \alpha = .005'};
str(5) = {' t = 0:900'};
set(gcf,'CurrentAxes',h)
text(.025,.6,str,'FontSize',12)
```

生成的图形如图 9-36 所示。

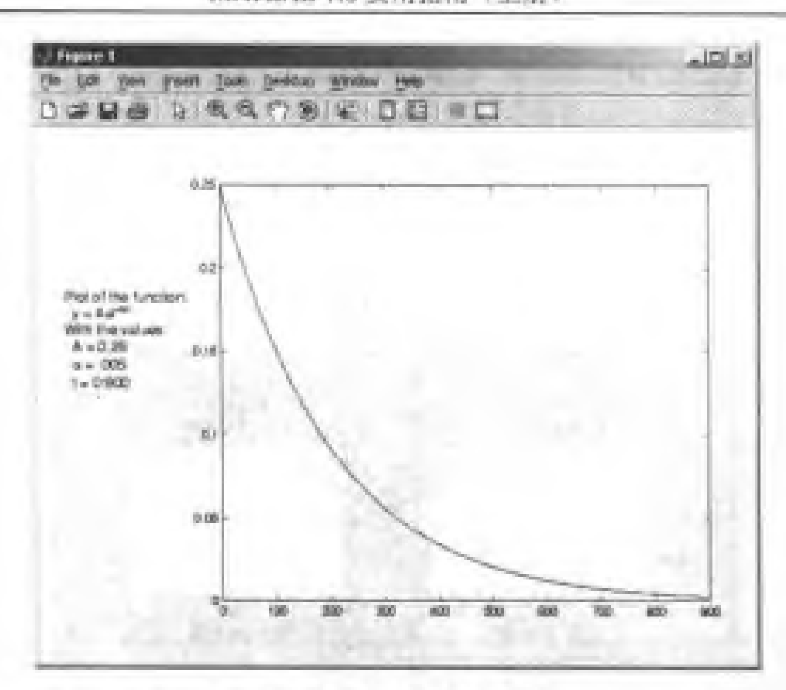


图 9-36　在图中显示文本

可以创建多个坐标轴，在不改变数据的情况下用不同比例显示图形对象，例如，

```
h(1) = axes('Position',[0 0 1 1]);
sphere
h(2) = axes('Position',[0 0 .4 .6]);
sphere
h(3) = axes('Position',[0 .5 .5 .5]);
sphere
h(4) = axes('Position',[.5 0 .4 .4]);
sphere
h(5) = axes('Position',[.5 .5 .5 .3]);
sphere
set(h,'Visible','off')
```

生成图 9-37。

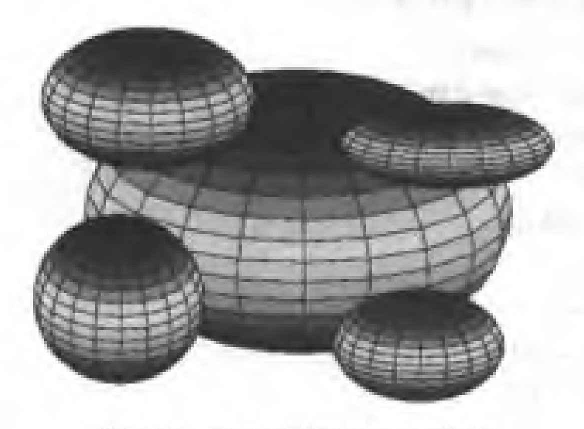


图 9-37　用不同比例显示图形对象

图 9-37 中，每个圆球用相同的数据定义。但是，因为父坐标系占据了大小、位置不同的区域，这些圆球看起来具有不同的大小和形状。

### 9.15.4　单个坐标轴的控制

创建图形时，MATLAB 会自动控制坐标轴的范围、标记位置和标记的标签。但是，可以通过设置合适的属性来手工指定这些值。

给模式控制的属性（如 XLim 属性有一个相连的 XLimMode 属性）指定值时，MATLAB 会将模式设置为手工方式，并覆盖自动指定方式。因为这些模式属性的默认值是自动的，调用高级函数如 plot 或 surf 等会将这些模式重置为 auto。

表 9-5 描述了这些属性。

**表 9-5　模式属性**

| 属　性 | 功　能 |
| --- | --- |
| XLim,YLim,ZLim | 设置坐标轴对应的值的范围 |
| XLimMode, YLimMode, ZLimMode | 指定坐标轴对应的值的范围是由 MATLAB 自动确定还是由用户手工指定 |
| XTick, YTick, ZTick | 设置沿坐标轴分布的标记的位置 |
| XTickMode, YTickMode, ZTickMode | 指定坐标轴上标记的位置是由 MATLAB 自动确定还是由用于手工指定 |
| XTickLabel, YTickLabel, ZTickLabel | 为标记指定标签 |
| XTickLabelMode, YTickLabelMode, ZTickLabelMode | 指定标记的标签是由 MATLAB 自动确定还是由用户手工指定 |
| XDir,YDir,ZDir | 设置坐标轴上值的增长方向 |

**1．设置坐标轴对应的值的范围**

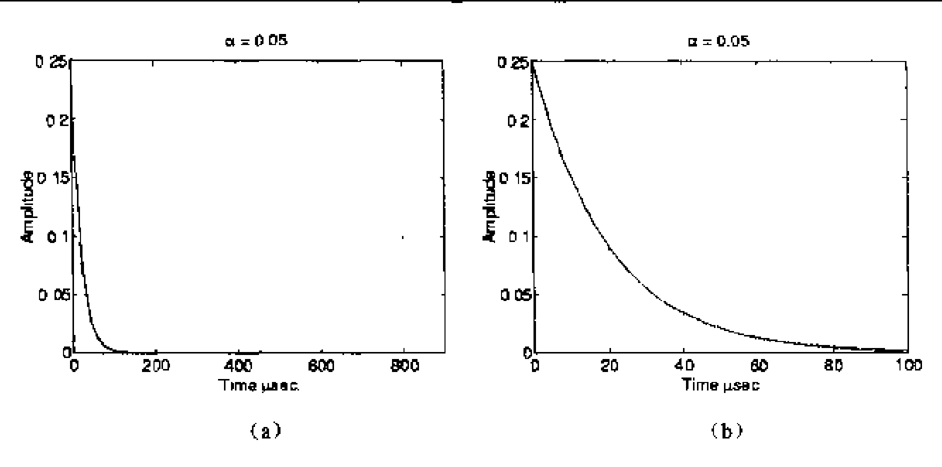
MATLAB 基于数据范围自动确定每个坐标轴上值的范围，可以通过指定 XLim,YLim 或 ZLim 来覆盖这些选定的范围。例如，绘制函数 $Ae^{-\alpha t}$ 的图形，其中 $A$=0.25, $\alpha$ =0.05, $t$=0~900。

```
t = 0:900;
plot(t,0.25*exp(-0.05*t))
```

图 9-38 中，图（a）显示了绘制结果。MATLAB 会选择在 $x$ 和 $y$ 轴方向上把数据压缩到坐标轴范围内。但是，因为图形包含少量 $t$=100 秒以外的信息，改变 $x$ 轴的限制可以改进图形的有用性。如果坐标系的句柄是 axes_handle，则下面的语句：

```
set(axes_handle,'XLim',[0 100])
```

创建图 9-38（b）。




（a）　　（b）

图 9-38　设置坐标轴上值的范围

**2．半自动化设置坐标轴上值的范围**

可以指定坐标系范围限制的最小值或最大值，并允许未指定端自动调整大小，可以手工设置一个上限或下限，与 Inf 或-Inf 一起设置范围，例如，

```
set(axes_handle,'XLim',[0 Inf])
```

把 XLimMode 属性设置为 auto 并允许 MATLAB 确定 XLim 的最大值。近似地，下面的语句将 XLimMode 属性的值设置为 auto 并允许 MATLAB 确定 XLim 的最小值。

```
set(axes_handle,'XLim',[-Inf 800])
```

MATLAB 基于数据范围选择标记位置并生成等间隔的标记。可以用 XTick, YTick 和 ZTick 属性指定其他标记。

例如，如果值 0.075 是函数 $Ae^{-\alpha t}$ 的图形上感兴趣的点，可以通过指定标记来包括这些值。

```
set(gca,'YTick',[0 0.05 0.075 0.1 0.15 0.2 0.25])
```

结果如图 9-39 中所示。

可以用 XTickLabel,YTickLabel 和 ZTickLabel 属性把标记标签从数值改为字符串。例如，用字符串 Cutoff 标注 y 轴上的值 0.075，可以通过用“|”字符间隔每个标签来把所有 y 轴标签指定为一个字符串。

```
set(gca,'YTickLabel','0|0.05|Cutoff|0.1|0.15|0.2|0.25')
```

效果如图 9-40 所示。

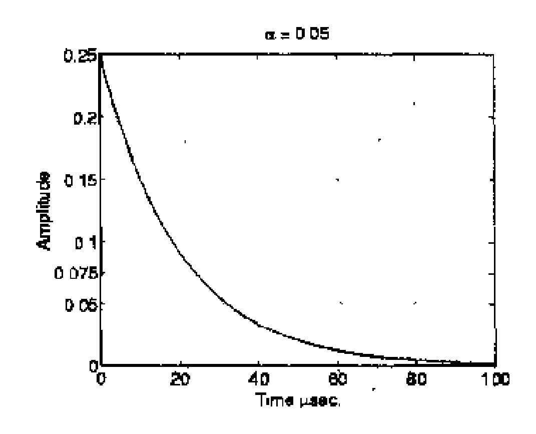



图 9-39　在图上标记感兴趣的点

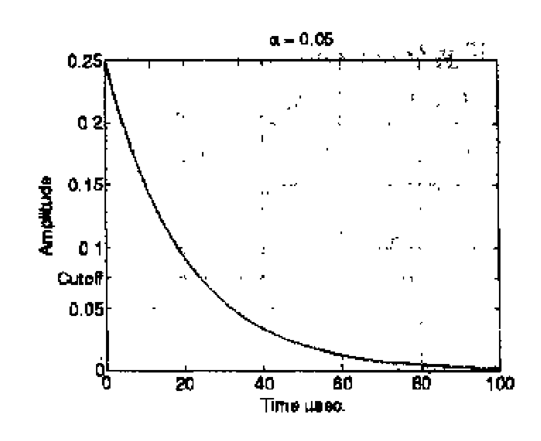



图 9-40　指定标记标签

### 3. 改变坐标轴的方向

Xdir,Ydir 和 Zdir 属性控制各坐标轴上数值递增的方向。在默认的二维视图中，$x$ 轴的值从左向右递增，$y$ 轴的值从下向上递增，$z$ 轴的值向外递增。

可以通过将 XDir,YDir 和 ZDir 属性的值设置为 reverse 来改变数值递增的方向。例如，下面将 XDir 属性的值设置为 reverse。

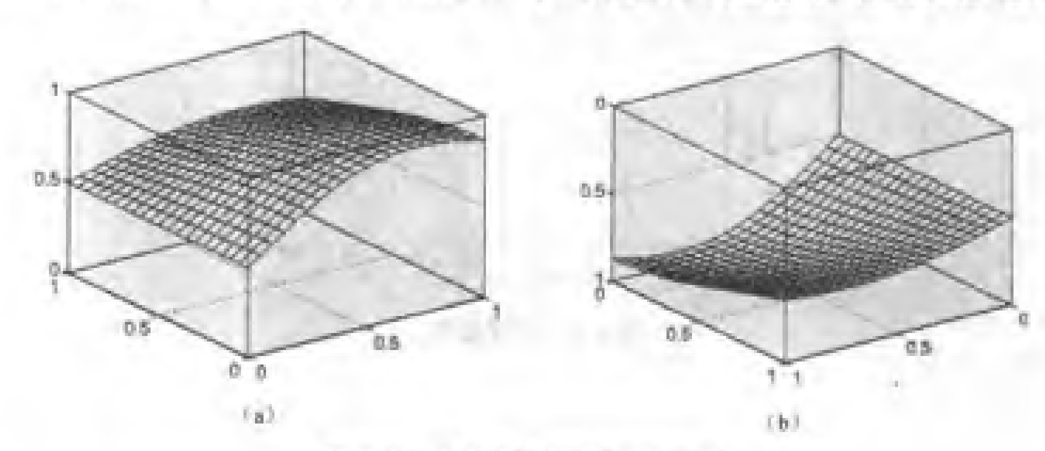
```
set(gca,'XDir','reverse')
```

效果如图 9-41 所示，$x$ 轴的数值从左向右递减。




图 9-41　反转 $x$ 轴

在三维视图中，$y$ 轴数值从前向后递增，$z$ 轴数值从下向上递增，如图 9-42（a）所示。



（a）　（b）

图 9-42　反转坐标轴前后的曲面显示

将 $x$, $y$ 和 $z$ 轴对应的数值反转：

```
set(gca,'XDir','rev','YDir','rev','ZDir','rev')
```

效果如图 9-42（b）所示。

## 9.15.5　使用多个 $x$ 轴和 $y$ 轴

XAxisLocation 和 YAxisLocation 属性指定在图形的哪一侧放置 $x$ 轴和 $y$ 轴。可以通过强制添加两个坐标系对象并使用 XAxisLocation 和 YAxisLocation 属性在图形的两侧放置坐标轴来创建两个不同的 $x$ 轴和 $y$ 轴。

下面的例子在一幅图中显示两套数据，用底部和左侧的边作为一套数据图形的 $x$ 轴和 $y$ 轴，顶部和右侧的边作为另一套数据图形的 $x$ 轴和 $y$ 轴。

```
hl1 = line(x1,y1,'Color','r');
ax1 = gca;
set(ax1,'XColor','r','YColor','r')
```

使用 line 和 axis 函数可以很容易地实现图形的添加，例如，

```
ax2 = axes('Position',get(ax1,'Position'),...
          'XAxisLocation','top',...
          'YAxisLocation','right',...
          'Color','none',...
          'XColor','k','YColor','k');
```

用与 $x$ 轴和 $y$ 轴相同的颜色绘制第二套数据的图形：

```
hl2 = line(x2,y2,'Color','k','Parent',ax2);
```

生成图 9-43。

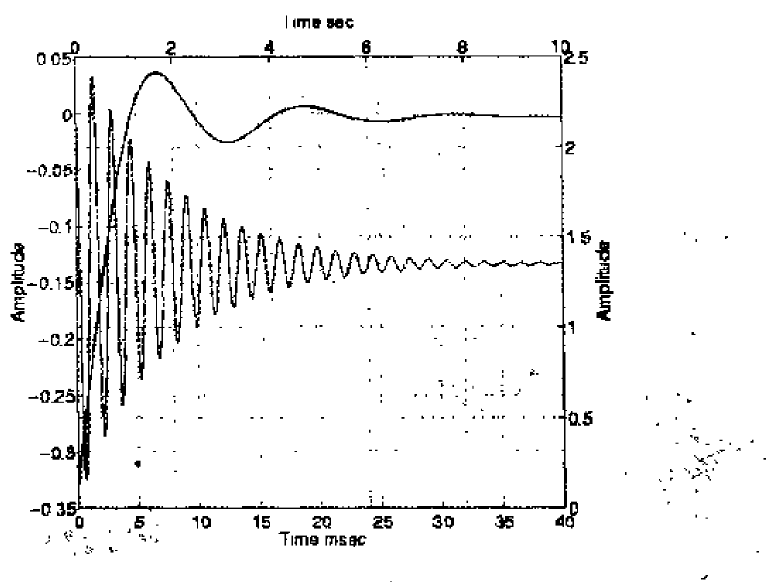



图 9-43　绘多轴图

# 第 10 章　定制二维图形

MATLAB 具有强大的图形表达功能，它不仅提供了很多实用的专业图形绘制函数，还提供了可以绘制基本图元的句柄图形对象创建函数，从而可以像 VC,VB 那样进行更自由的图形开发工作。本章介绍绘制 MATLAB 二维图形的基本图形元素以及如何利用这些元素实现二维图形的定制。

## 10.1　基本图形元素

任何复杂的图形都是由简单的图形元素组成的，最基本的图形元素包括多义线、椭圆弧、自由曲线和文本。前两种还可以衍生出直线段、矩形、圆角矩形、圆、圆弧和区域等。下面逐一进行介绍。

### 10.1.1　直线段、多义线和曲线——Line 对象

MATLAB 中的直线段和多义线用 Line 对象表示。用 line 函数创建直线段对象。按照直线逼近的思路，还可以用该函数创建曲线。

该函数的语法格式为

```
line(X,Y)
line(X,Y,Z)
line(X,Y,Z,'PropertyName',PropertyValue,...)
line('PropertyName',PropertyValue,...)
h = line(...)
```

各语法格式可作如下描述：

- line(X,Y)　在当前坐标系中添加向量 $X$ 和 $Y$ 定义的直线段。如果 $X$ 和 $Y$ 是大小相同的矩阵，则 line 函数用每个列的数据绘制一条直线段。
- line(X,Y,Z)　创建三维直线段。
- line(X,Y,Z,'PropertyName',PropertyValue,...)　用属性名/属性值指定的值创建直线段，没有指定的属性取默认值。
- line('XData',x,'YData',y,'ZData',z,'PropertyName',PropertyValue,.. .)　在当前坐标系中创建直线段，将属性值定义为变量。是 line 函数的低级形式，不接受矩阵坐标数据。
- h = line(...)　返回句柄列向量，向量值对应于创建的每个 Line 对象。

表 10-1 中列出了 Line 对象的部分属性，通过设置属性，可以改变 Line 对象的位置和外观。

表 10-1　Line 对象的部分属性

| 属　性 | 说　明 | 属 性 值 |
|---|---|---|
| XData | 直线段顶点的 x 坐标值 | 值为向量或矩阵，默认值为[0 1] |
| YData | 直线段顶点的 y 坐标值 | 值为向量或矩阵，默认值为[0 1] |
| Zdata | 直线段顶点的 z 坐标值 | 值为向量或矩阵，默认值为[ ] |
| LineStyle | 直线段的线型 | 值为 -、--、:、-.、none 中的一种，默认值为- |
| LineWidth | 直线段的线宽 | 值为标量，默认值为 0.6 磅 |
| Marker | 直线段顶点的标记 | 值为+、*、.、x、s、d、^、v、>、<、p、h 和 none 中的一种，默认值为 none |
| MarkerEdgeColor | 非填充标记的颜色或填充标记的边的颜色 | 值为指定的颜色，none 或 auto，默认值为 auto |
| MarkerFaceColor | 填充标记的内部颜色 | 值为指定的颜色，none 或 auto，默认值为 none |
| MarkerSize | 标记的大小 | 值为磅表示的大小，默认值为 6 |

下面结合几个例子介绍 line 函数的使用。

首先创建一条宽度为 5 磅的直线段，起点为[1 20]，终点为[1 30]。

```
X1=[1 20];
Y1=[1 30];
line(X1,Y1,'LineWidth',5)
```

生成图 10-1。

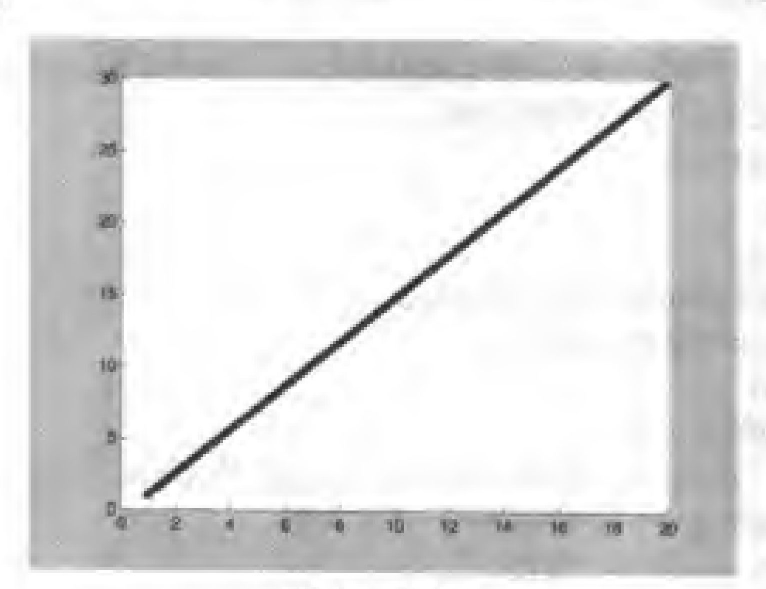



图 10-1　直线段

利用 line 函数还可以生成多义线，如下所示：

```
X2=[1 9 20 28];
Y2=[1 25 10 32];
line(X2,Y2,'LineStyle','-','Marker','+')
```

生成图 10-2。多义线的顶点用“+”标记进行了标注。

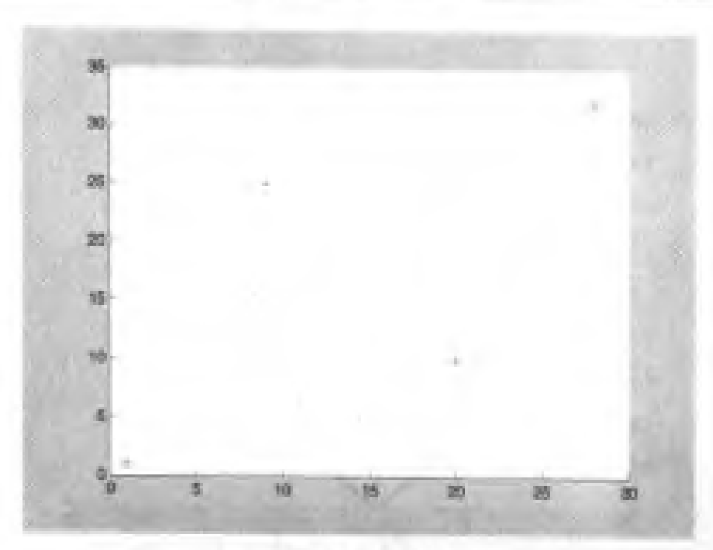


图 10-2　多义线

如果 line 函数中的位置参数为矩阵，则可以同时生成多条直线段或多义线，在命令窗口中键入下面的命令行：

```
X3=[1 1 1;20 20 20;28 28 28];
Y3=[1 5 3;10 15 9;32 30 35];
line(X3,Y3)
```

生成图 10-3。

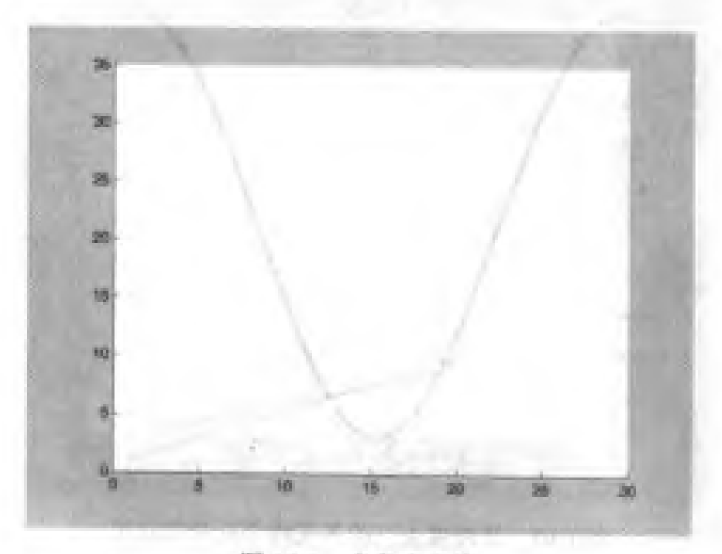


图 10-3　多条多义线

下面用多条直线段逼近余弦曲线，用下面的代码将[0,2π]范围内的余弦曲线用有 11 个顶点的多义线表示。

```
t=0:pi/5:2*pi;
line(t,cos(t),'LineWidth',3)
```

生成图 10-4。曲线显然不平滑。

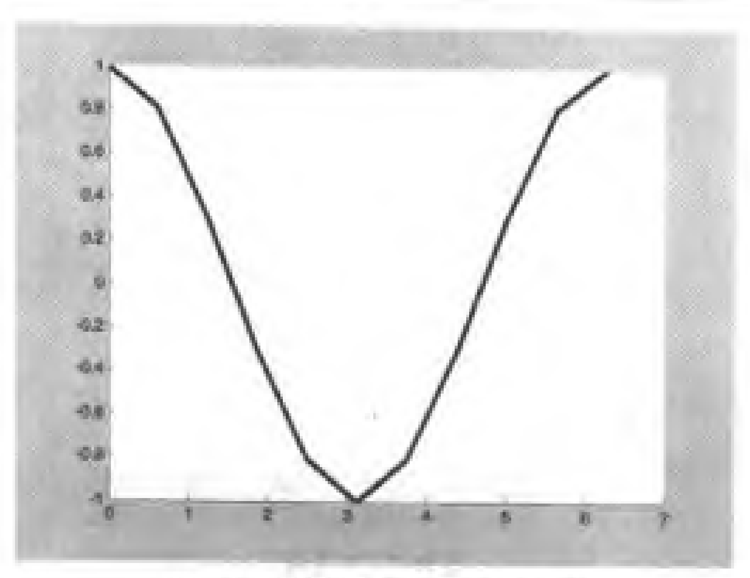



图 10-4　用多义线近似表示余弦曲线

下面用精度更高的多义线（21 个顶点）近似表示。

```
t=0:pi/20:2*pi;
line(t,cos(t),'LineWidth',3)
```

生成图 10-5。现在比较平滑了。

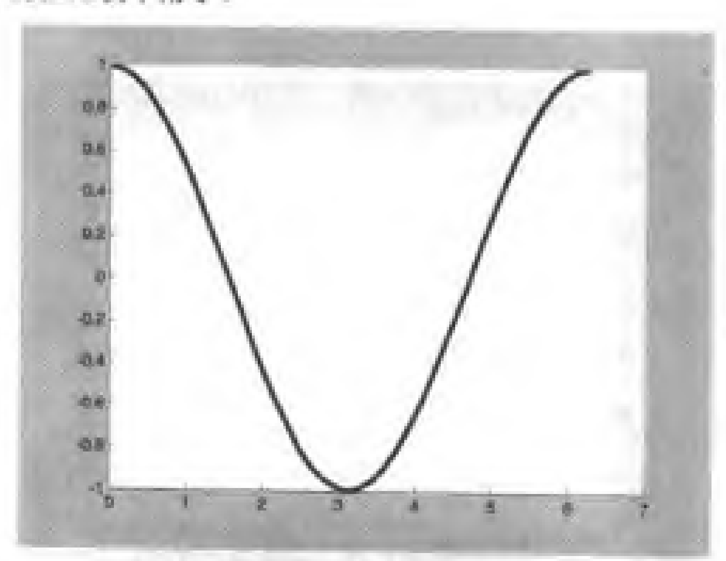



图 10-5　用精度更高的多义线逼近余弦曲线

### 10.1.2　矩形、圆角矩形、椭圆、圆及对应的区域图形——Rectangle 对象

MATLAB 中，矩形、圆角矩形、椭圆、圆及对应的区域图形都用 Rectangle 对象表示。用 rectangle 函数可以创建 Rectangle 对象。其语法格式为

```
rectangle
rectangle('Position',[x,y,w,h])
rectangle(...,'Curvature',[x,y])
h = rectangle(...)
```

各语法格式可作如下描述：

- rectangle　绘一个单位正方形，左下角坐标为[0 0]，右上下角坐标为[1 1]，曲率为[0,0]（即没有曲率）。
- rectangle('Position',[x,y,w,h])　给定起点[x, y]、矩形的宽 $w$ 和高 $h$，绘制矩形。
- rectangle(...,'Curvature',[x,y])　给定矩形边的曲率，$x$ 为水平曲率，$y$ 为垂直曲率。值为[0 0]时，创建矩形；值为[1 1]时，创建椭圆。如果只指定一个值，则水平边和垂直边都会发生相同长度的弯曲。曲率的大小由短的边确定。
- h = rectangle(...)　返回所创建的矩形对象的句柄。

下面结合两个实例进行介绍。

在同一个坐标系中创建矩形、圆角矩形、椭圆和圆各一个，并使用了不同的线型和线宽。

```
rectangle('Position',[1,1,20,10], 'LineWidth',3);
rectangle('Position',[5,3,10,15], 'Curvature',[1 1]);
rectangle('Position',[5,3,10,10], 'Curvature',[1 1], 'LineWidth',3);
rectangle('Position',[4,5,12,8], 'Curvature',.4,'LineStyle', '--');
axis equal
```

生成图 10-6。

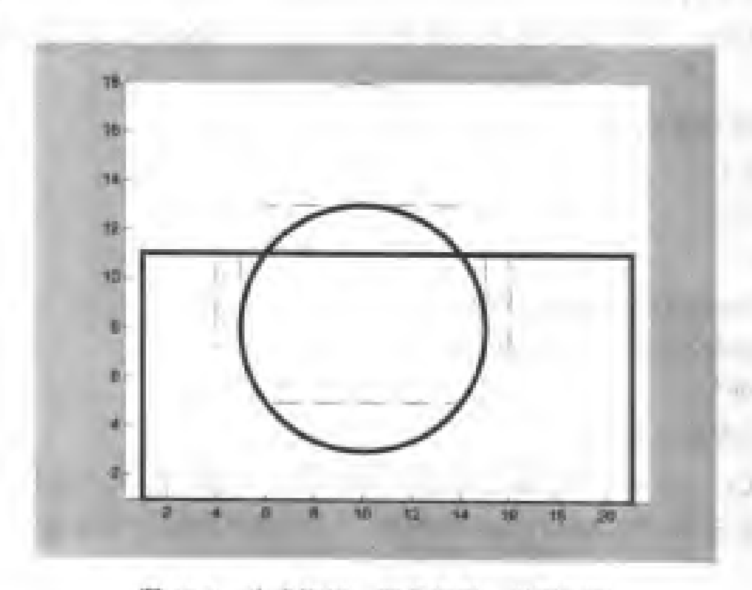


图 10-6　生成矩形、圆角矩形、椭圆和圆

利用 rectangle 函数还可以生成对应的区域，如下所示。键入相关命令行：

```
rectangle('Position',[5,3,10,15], 'Curvature',[.8 .4], 'FaceColor', 'g');
rectangle('Position',[5,3,10,10], 'Curvature',[1 1], 'LineWidth',3, 'FaceColor', 'r');
axis equal
```

生成图 10-7。

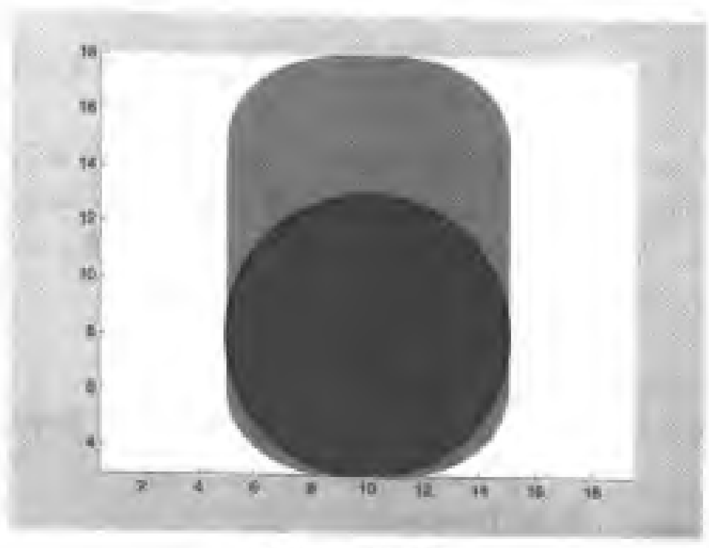


图 10-7　生成区域

### 10.1.3　多边形——Patch 对象

任意多边形及多边形区域是用 Patch 对象表示的，该对象由 patch 函数创建。这里只讨论二维的情况，在第 11 章中还要讨论三维的情况，关于 Patch 对象和 patch 函数有更多更精彩的描述。

patch 函数的语法形式为

```
patch(X,Y,C)
patch(X,Y,Z,C)
patch(FV)
patch(...'PropertyName',PropertyValue...)
patch('PropertyName',PropertyValue...)
handle = patch(...)
```

各语法格式可作如下描述：

- patch(X,Y,C)　在当前坐标系中添加二维面片。$X$ 和 $Y$ 的元素共同指定多边形的顶点。如果 $X$ 和 $Y$ 是矩阵，则 MATLAB 利用每一列的数据绘制一个多边形。$C$ 确定多边形的颜色。
- patch(X,Y,Z,C)　在三维坐标系中创建面片。
- patch(FV)　用结构 FV 创建面片，该结构包含有字段 vertices、faces 和可选的 facevertecdata。这些字段会生成对应的 Vertices、Faces 和 FaceVertexCData 属性值。
- patch(...'PropertyName',PropertyValue...)　指定顶点坐标以后用属性名/属性值匹配对指定其他面片属性。
- patch('PropertyName',PropertyValue...)　完全用属性名/属性值的形式指定属性值。使用这种方式，可以忽略颜色的指定，因为除非给 FaceColor 和 EdgeColor 属性指定一个值，MATLAB 会使用默认的小面颜色和边线颜色。使用这种形式可以用 Faces 和 Vertices 属性定义面片。

- handle = patch(...)　返回所创建的面片对象的句柄。

下面用顶点/小面表示方法和顶点/颜色表示方法创建了一个等腰三角形和一个多边形区域。

```
vert=[1 2;3 2;2 4];
face=[1 2 3];
patch('Faces',face, 'Vertices',vert, 'FaceColor', 'none');
x = [3;4;5;6;6.5;4];
y = [4;2.5;4;3;4;6];
tcolor(1,1:6) = [.7 .7 .7 .7 .7 .7];
patch(x,y, tcolor);
axis([1 7 1 7])
```

生成图 10-8。

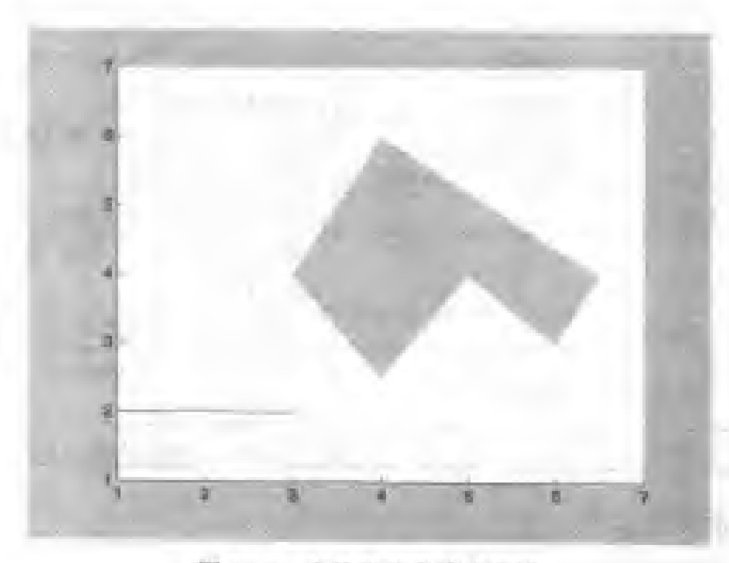



图 10-8　多边形和多边形区域

## 10.1.4　文本——Text 对象

文本用 Text 对象表示，该对象由 text 函数创建。函数语法如下所示：

```
text(x,y,'string')
text(x,y,z,'string')
text(...'PropertyName',PropertyValue...)
h = text(...)
```

各语法格式可作如下描述：

- text　是创建文本图形对象的低级函数。使用它将文本字符串放在指定的位置上。
- text(x,y,z,'string')　在位置($x$, $y$)上添加字符串。
- text(...'PropertyName',PropertyValue...)　指定字符串的属性。
- text('PropertyName',PropertyValue...)　完全忽略坐标，用属性名/属性值数据对指定所有属性。
- h = text(...)　返回包含文本对象的列向量，每个对象一个句柄。

表 10-2 列出了 Text 对象的部分属性。设置对象属性，可以改变文本的位置、大小、字体和样式等。

表 10-2　Text 对象的部分属性说明

| 属　性 | 说　明 | 属 性 值 |
|---|---|---|
| Extent | 文本对象的位置和大小 | 值为[left, bottom, width, height] |
| HorizontalAlignment | 文本字符串的水平对齐方式 | 值为 left, center, right，默认值为 left |
| Position | 文本包围矩形的位置 | 值为[x, y, z]坐标，默认值为[] |
| Rotation | 文本对象的方向 | 值为标量，默认值为 0 |
| Units | Extent 和 Position 属性值的单位 | 值为 pixels, normalized, inches, centimeters, points 和 data，默认值为 data |
| VerticalAlignment | 文本字符串的垂直对齐方式 | 值为 top, cap, middle, baseline 和 bottom，默认值为 middle |
| BackgroundColor | 文本包围矩形的颜色 | 值为 ColorSpec，默认值为 none |
| EdgeColor | 文本包围矩形边的颜色 | 值为 ColorSpec，默认值为 none |
| LineWidth | 文本包围矩形边的线宽 | 值为标量，默认值为 0.5 |
| LineStyle | 文本包围矩形边的线型 | 值为-、--、:、-.和 none，默认值为- |
| Margin | 文本至包围边的距离，单位为像素 | 值为标量，默认值为 2 |
| FontAngle | 选择斜体 | 值为 normal、italic 和 oblique，默认值为 normal |
| FontName | 选择字体 | 值为系统支持的一种字体，或字符串 FixedWidth，默认值为 Helvetica |
| FontSize | 字体大小 | 值为 FontUnits 属性值的大小，默认值为 10 磅 |
| FontUnits | FontSize 属性值的单位 | 值为 points、normalized、inches、centimeters 和 pixels，默认值为 points |
| FontWeight | 文本字符的粗度 | 值为 light、normal、demi 和 bold，默认值为 normal |

下面创建多种形式的文本。

```
text(10,30, '添加文本 1');
text(10,25, '添加文本 2', 'FontSize',20, 'FontAngle', 'Italic');
text(10,20, '添加文本 3', 'FontName', '黑体', 'FontWeight', 'bold');
text(10,10, '添加文本 4', 'Rotation',45);
axis([5 20 8 35])
```

结果如图 10-9 所示。




图 10-9　添加文本

## 10.2　定制二维图形

前面介绍了在 MATLAB 中创建基本图形元素的方法。有了基本图形元素，就可以灵活地运用它们，形成更复杂的图形。MATLAB 中提供了很多专业图形的创建函数，现在有了基本图形元素，我们自己也可以把这些函数编写出来；不仅如此，还可以根据自己的需要，创建自己需要的图形。

下面创建一个最简单的面积图，它要求输入数据只能是顶点位置的横坐标向量和纵坐标向量，不能是矩阵。但是它可以显示顶点坐标。

首先创建一个M文件myarea.m，代码如下所示：

```
function myarea(x,y,label)
  len=length(x);
  miny=min(y);
  verts(1,1)=x(1);
  verts(1,2)=miny-1;
  for i=1:len
      verts(i+1,1)=x(i);
      verts(i+1,2)=y(i);
  end
  verts(len+2,1)=x(len);
  verts(len+2,2)=miny-1;
  faces=[1:len+2];
  patch('Faces',faces,'Vertices',verts,'FaceColor','g');
  if label==1
      for i=1:len
          text(x(i),y(i),['(',num2str(x(i)),',',num2str(y(i)),')']);
      end
  end
  axis auto
```

然后在命令窗口中键入下面的命令行，绘制一个没有顶点标注的面积图和一个有顶点标注的面积图。

```
x=[1 2 3 4];
y=[10 5 20 13];
myarea(x,y,0);
figure;myarea(x,y,1);
```

生成图10-10和图10-11。

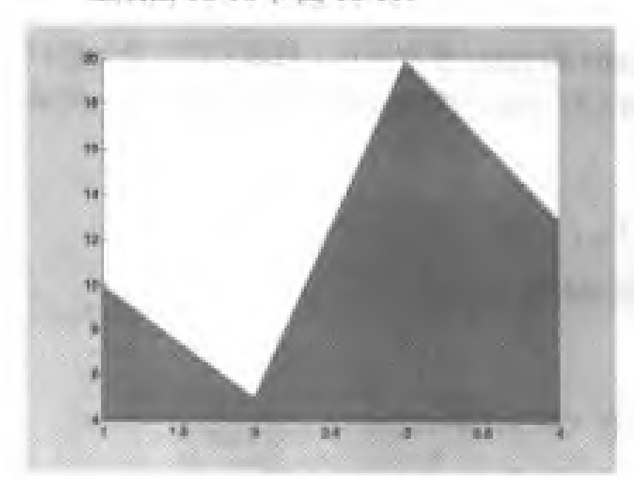


图10-10　没有顶点标注的面积图

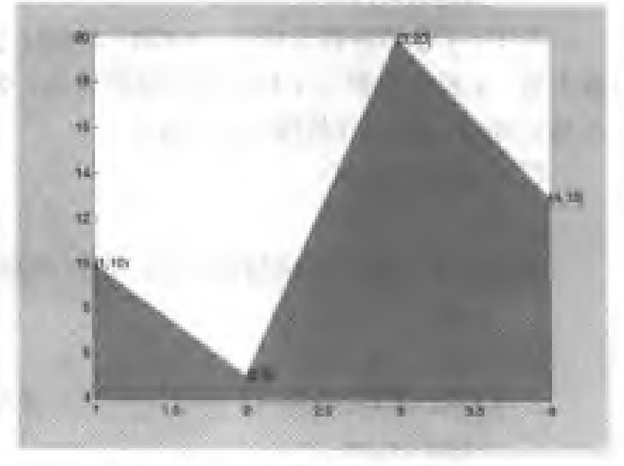


图10-11　有顶点标注的面积图

# 第 11 章　三维模型的建立

第 7 章和第 10 章我们讨论了用 MATLAB 进行二维图形绘制的方法和手段。但真实世界里面的对象大部分是三维的，如何用三维几何形体表现真实场景，如何使科学计算数据的图形表现更真实更生动，如何模拟虚拟场景等，一直是计算机图形学试图并正在解决的问题。但三维问题比二维平面问题要复杂得多，如果从底层开发，将要面临很多困难。MATLAB 在这方面提供了一个很高的平台，它将处理场景所必需的图元、光照等封装到很少的几个对象中，直接利用这些对象，结合 MATLAB 提供的工具函数，可以很快建立三维模型，并进行着色、光照、材质和交互等处理。本章主要介绍建模。

## 11.1　线形模型的建立

除了空间点以外，线形模型就是最简单的模型了。下面介绍几种线形模型的建立方法，包括参数曲线、样条曲线、用矩阵数据绘图和三维等值线图、三维向量图的绘制。

### 11.1.1　参数曲线

使用 ezplot 函数和 ezplot3 函数，可以很方便地绘制二维和三维参数曲线；使用 ezpolar 函数，可以在极坐标中绘制参数曲线。下面主要介绍前两个函数的使用。

用 ezplot 函数绘制给定一元或二元函数的曲线。函数形式可以是一般形式，也可以是参数形式。以二元函数为例，函数调用格式为

```
ezplot(f,[xmin,xmax,ymin,ymax])
ezplot(x,y)
ezplot(x,y,[tmin,tmax])
```

其中，$f$ 表示函数表达式，xmin，xmax，ymin 和 ymax 分别表示 $x$ 值和 $y$ 值的最小值和最大值。$x$ 和 $y$ 参数为 $x$ 和 $y$ 的参数表达式，tmin 和 tmax 为参数 $t$ 的最小值和最大值。如果不指定范围，默认时范围为 0～2π。

假设函数形式为

$$x^2-y^2=1$$

在命令窗口键入下面的命令行，可以绘制该函数的曲线图。

```
ezplot('x^2-y^2-1')
```

结果如图 11-1 所示。

三维参数曲线用 ezplot3 函数绘制。该函数有 3 种语法格式：

```
ezplot3(x,y,z)
ezplot3(x,y,z,[tmin,tmax])
ezplot3(...,'animate')
```

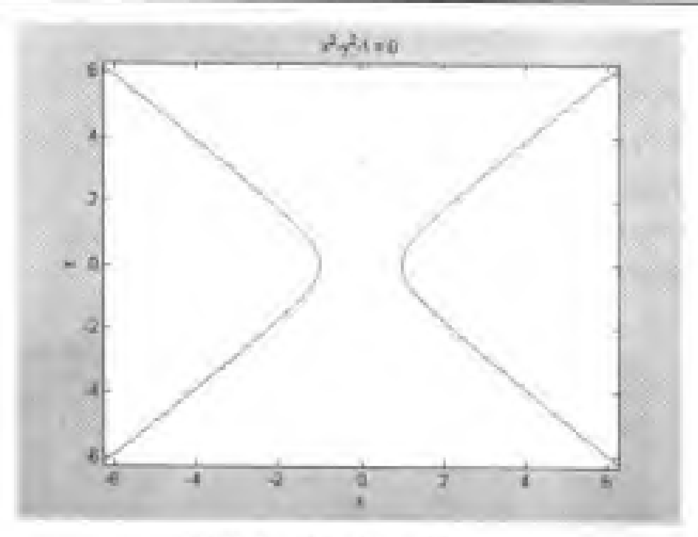


图 11-1　函数的曲线图

其中，曲线用 $x = x(t)$, $y = y(t)$和 $z = z(t)$ 的形式表示，默认时，参数取值范围为 $0 < t < 2\pi$。用 tmin 和 tmax 可以指定参数范围，指定 animate 参数，可以用动画形式生成曲线。

假设有下面的参数方程：

$$\begin{cases} x = e^{\frac{t}{10}} \\ y = \sin t \cdot \cos t \\ z = t \end{cases}$$

在命令窗口键入下面的命令行，可以绘制该方程的曲线图。

```
ezplot3('exp(t/10)','sin(t)*cos(t)','t',[0,6*pi])
```

生成图 11-2。

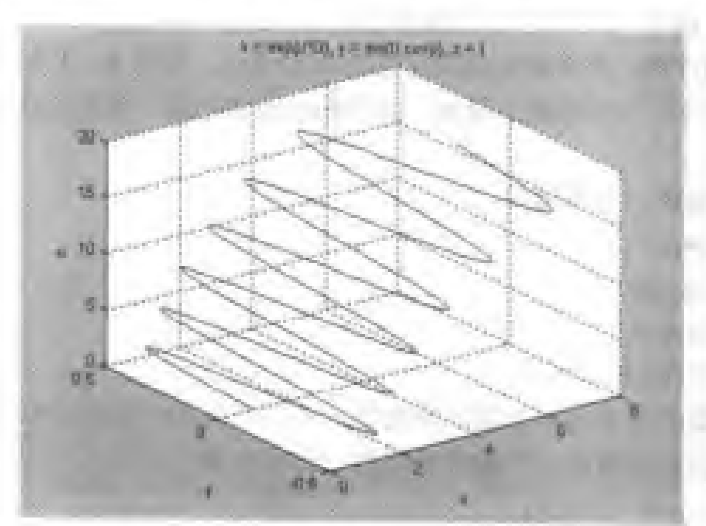


图 11-2　三维参数曲线

## 11.1.2　样条曲线

利用 MATLAB 的样条工具箱，可以绘制多种样条曲线。关于样条工具箱的使用，可以

参见文献 5。

在命令窗口键入下面的代码，可以创建一条 B 样条曲线。

```
points = [0 0 1 1 0 -1 -1 0 0 ;0 0 0 1 2 1 0 -1 -2];
plot(points(1,:),points(2,:),':')
values = spcrv(points,3);
hold on, plot(values(1,:),values(2,:)), hold off
```

生成图 11-3。

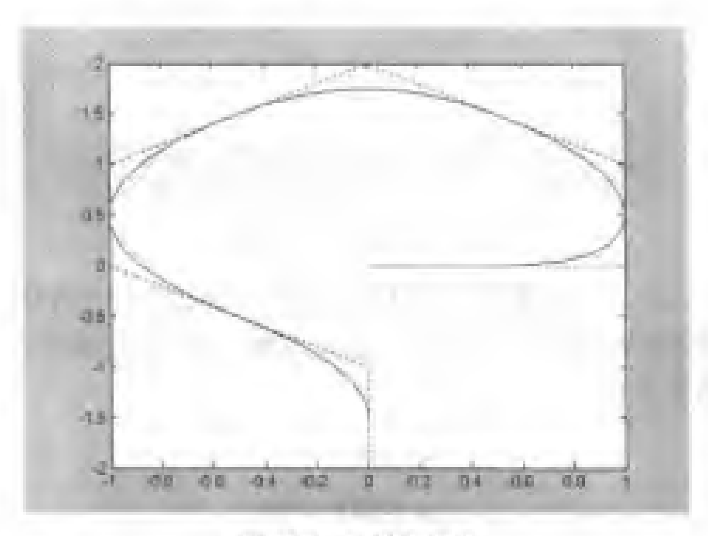


图 11-3　B 样条曲线

### 11.1.3　用给定数据绘图

第 2 章介绍了 line 函数，知道利用该函数可以绘制三维线形图。除了 line 函数外，MATLAB 还提供了 plot3 函数绘制三维线形图。

利用 plot3 函数绘图，需要指定线条穿过的点的坐标。假设 ***X***、***Y*** 和 ***Z*** 是 3 个向量，分别表示一系列点的 $x$ 坐标、$y$ 坐标和 $z$ 坐标，则下面的命令创建一条经过这些点的三维曲线。

```
plot3(X,Y,Z)
```

下面的命令行创建一条螺旋线。

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
axis square; grid on
```

生成的线形图如图 11-4 所示。

如果 plot3 函数的变量为大小相同的矩阵，则用它们的列数据分别绘制一条三维线条。下面的命令行绘制给定范围内一个数学函数的三维线形图。

```
[X,Y] = meshgrid([-2:0.1:2]);
Z = X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z);
grid off
```

效果如图 11-5 所示。

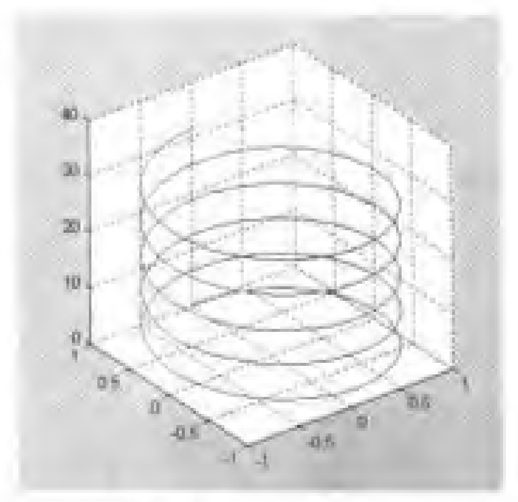




图 11-4　三维线形图

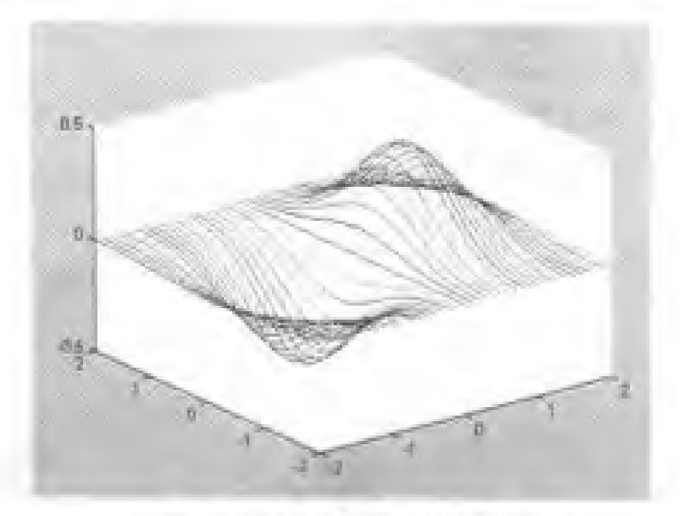


图 11-5　参数为矩阵时的绘图效果

线形图虽然比较简单，但如果灵活运用，也能有丰富的表现力。如图 11-5 就可以表现面的信息。另外，在科学计算方面，常用到等值线图和矢量图。第 2 章介绍了二维等值线图和矢量图，实际上，MATLAB 还提供了绘制三维等值线图和三维向量图的函数。下面分别予以介绍。

### 11.1.4　三维等值线图

用 contour3 函数绘三维等值线图，其调用格式为：

- contour3(Z)　绘矩阵 **Z** 的三维等值线图。**Z** 可以看作是 $x$-$y$ 平面以上的高程，它必须至少是一个 2×2 的矩阵。等值线的水平数和等值线水平值自动选择。$x$ 轴和 $y$ 轴的范围分别为[1:n]和[1:m]，其中[m,n] =size(Z)。
- contour3(Z,n)　绘 **Z** 矩阵的具有 $n$ 个水平的三维等值线图。
- contour3(Z,v)　绘矩阵 **Z** 的三维等值线图，等值线位于向量 $v$ 指定的值处，等值线的水平数等于 length(v)。使用 contour(Z,[i i])，可以只绘水平 $i$ 的等值线。
- contour3(X,Y,Z), contour3(X,Y,Z,n)和 contour3(X,Y,Z,v)　使用 ***X*** 和 ***Y*** 定义 $x$ 轴和 y 轴的界限。如果 ***X*** 为一矩阵，则 X(1,:)定义 $x$ 轴；如果 ***Y*** 为一矩阵，则 Y(:,1) 定义 y 轴。当 ***X*** 和 ***Y*** 均为矩阵时，它们必须与 ***Z*** 具有相同的大小，此时它们将指定一个表面。
- contour3(...,LineSpec)　用 LineSpec 指定的线型和颜色绘等值线。
- [C,h] = contour3(...)　返回等值线矩阵 ***C*** 和包含图形对象句柄的列向量。除非指定 LineSpec 属性（此时 contour3 函数创建直线图形对象），contour3 函数将创建阴影图形对象。

注意：如果没有指定 LineSpec 属性，colormap 和 caxis 将控制颜色。如果 ***X*** 或 ***Y*** 的间隔不确定，则 contour3 函数用确定间隔的等值线网格计算等值线，然后将数据转换到 ***X*** 或 ***Y***。

下面创建一个函数的三维等值线图。

```
[X,Y,Z]=peaks(40);
contour3(X,Y,Z,30)
```

```
grid off
view(-15,25)
```

生成图 11-6。

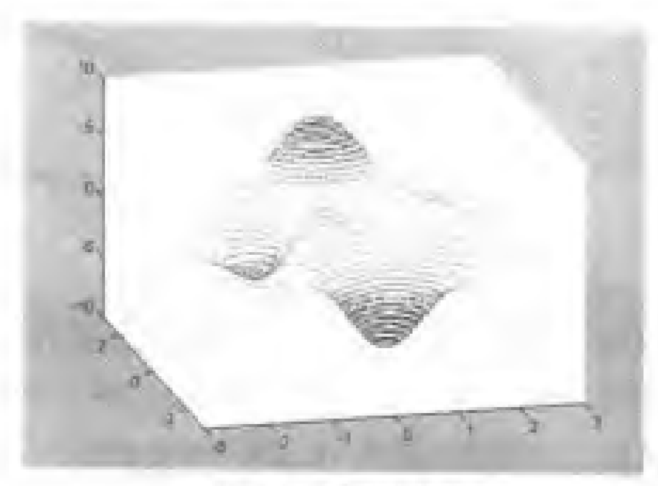


图 11-6　三维等值线图

## 11.1.5　三维向量图

用 quiver3 函数绘三维向量图。调用格式为：

- quiver3(Z,U,V,W)　在 **Z** 矩阵指定的等距表面点上绘向量图。quiver3 函数自动根据它们之间的距离确定显示比例，以防止显示溢出。
- quiver3(X,Y,Z,U,V,W)　指定点（*X, Y, Z*）处的元素（*U, V, W*），绘向量图。矩阵 ***X,Y,Z,U,V,W*** 必须大小相等，且包含对应位置和向量元素。
- quiver3(...,scale)　自动根据它们之间的距离确定显示比例，然后用比例值乘以原数据。
- quiver3(...,LineSpec)　用任意有效的 LineSpec 参数指定线型和颜色。
- quiver3(...,LineSpec,'filled')　填充 LineSpec 参数指定的标记。
- h = quiver3(...)　返回一个直线句柄向量。

下面绘给定函数的表面法向向量。

```
[X,Y,Z]=peaks(20);
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W,0.8);
hold on
surf(X,Y,Z);
colormap hsv
view(-35,45)
axis ([-3 3 -3 3 -6 8])
hold off
```

生成图 11-7。

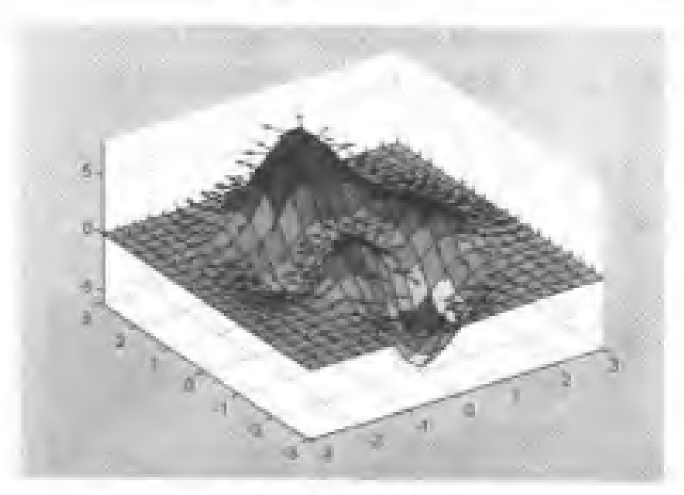


图 11-7　三维矢量图

## 11.2　曲面模型的建立

常见的三维曲面有参数曲面、样条曲面等。本节介绍如何绘制用函数表示的曲面、二次曲面、样条曲面和用给定数据绘制网格图、刻面图和曲面图。在后面的章节中，有时将刻面图和曲面图统称为表面图。

### 11.2.1　函数表示的曲面

用 ezmesh 函数绘制函数表示的网格。函数形式可以是一般形式，也可以是参数形式。该函数具有下面一些语法格式：

```
ezmesh(f,domain)
ezmesh(x,y,z)
ezmesh(x,y,z,[smin,smax,tmin,tmax]) 或 ezmesh(x,y,z,[min,max])
ezmesh(...,n)
ezmesh(...,'circ')
```

各语法格式可作如下描述：

- ezmesh(f)　创建函数 $f(x, y)$的图形，其中 $f$ 是一个字符串，表示两个变量的数学函数表达式。默认时，函数在$-2\pi< x<2\pi$，$-2\pi <y<2\pi$的范围内绘制。MATLAB 根据变量个数选择计算网格。
- ezmesh(f,domain)　在指定范围内绘制 $f$ 指定的函数的图形。范围可以是一个 4×1 的向量[xmin, xmax, ymin, ymax]，或者是 2×1 的向量[min, max]，分别表示 $x$，$y$ 和参数的范围。
- ezmesh(x,y,z)　绘制 $x = x(s,t)$，$y = y(s,t)$和 $z = z(s,t)$的参数曲面，范围为$-2\pi< s< 2\pi$，$-2\pi<t <2\pi$。
- ezmesh(x,y,z,[smin,smax,tmin,tmax])或 ezmesh(x,y,z,[min,max])　在指定范围内绘制参数曲面。
- ezmesh(...,n)　用 $n\times n$ 的网格在默认范围内绘制 $f$ 的曲面图，默认时 $n$ 等于 60。

- ezmesh(...,'circ')　在一个圆形区域内绘制 $f$ 的曲面图。

下面用 ezmesh 函数绘制一个函数的曲面图。假设函数为

$$z = y^2 - 3xy - x^2$$

在命令窗口键入下面的命令行：

```
ezmesh(' y^2-3*x*y-x^2')
```

生成图 11-8。

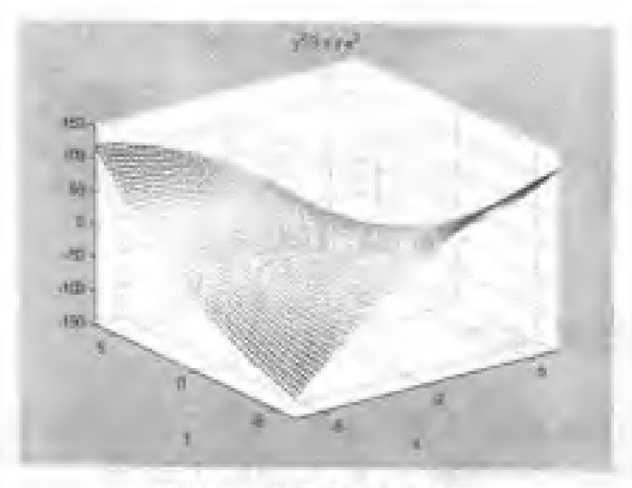


图 11-8　函数的三维网格图

用 ezsurf 函数绘制函数表示的曲面。该函数的语法格式为

```
ezsurf(f)
ezsurf(f,domain)
ezsurf(x,y,z)
ezsurf(x,y,z,[smin,smax,tmin,tmax])或 ezsurf(x,y,z,[min,max])
ezsurf(...,n)
ezsurf(...,'circ')
ezsurf(axes_handle,...)
h = ezsurf(...)
```

各语法格式可作如下描述：

- ezsurf(f)　创建函数 $f(x, y)$的图形，其中，$f$ 是变量 $x$ 和 $y$ 的函数。ezsurf 函数调用 surf 函数进行绘制。默认时，函数 $f$ 在定义域上 $-2\pi < x < 2\pi, -2\pi < y < 2\pi$ 绘图。MATLAB 根据变量个数选择计算网格。如果网格上某些点没有函数定义，则不绘制这些点。$f$ 参数可以是一个 M 文件函数或隐函数的句柄，也可以是一个字符串。
- ezsurf(f,domain)　在指定范围 domain 上绘制 $f$ 的图形。domain 可以是 4×1 的向量[xmin,xmax,ymin,ymax]或 2×1 的向量[min,max]（这里，min<x<max,min<y<max）。如果 $f$ 是变量 $u$ 和 $v$ 的函数，则绘图域的端点 umin,umax,vmin 和 vmax 按照字母顺序进行排序。这样，ezsurf('u^2 - v^3',[0,1],[3,6])在区域 $0 < u < 1, 3 < v < 6$ 上绘 $u^2 - v^3$ 的图形。
- ezsurf(x,y,z)　在区域 $-2\pi<s<2\pi, -2\pi<t<2\pi$ 上绘参数曲面 $x=x(s, t)$, $y=y(s, t)$和 $z=z(s, t)$的图形。
- ezsurf(x,y,z,[smin,smax,tmin,tmax])或 ezsurf(x,y,z,[min,max])　在指定区域上绘制参数曲面。

- ezsurf(…,n)　用 $n×n$ 的网格在默认区域上绘制函数 $f$。默认时 $n$=60。
- ezsurf(…, 'circ')　在区域中心的圆内绘函数 $f$ 的图形。
- ezsurf(axes_handle,…)　将图形绘制在 axes_handle 表示的坐标系内。
- h=ezsurf(…)　将一个曲面对象的句柄返回到 h 中。

将 ezmesh 函数的示例用于 ezsurf 函数，在命令窗口键入下面的命令行：

```
ezsurf(' y^2-3*x*y-x^2')
```

生成图 11-9。

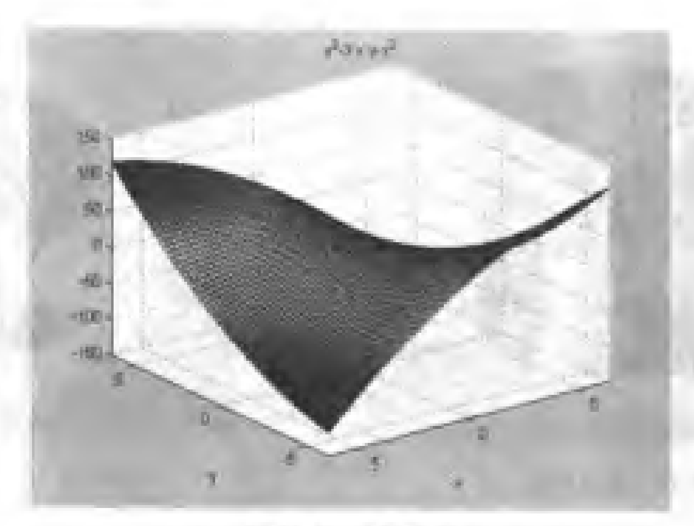


图 11-9　函数曲面

## 11.2.2　二次曲面

二次曲面是一种参数曲面，可以用一定的数学方程来表示。变换方程中的参数，可以改变曲面的形态。柱面、球面和锥面等是常见的二次参数曲面。MATLAB 提供了创建柱面和球面的函数。

### 1．柱面

用 cylinder 函数生成柱面。该函数的调用格式为：

- cylinder　生成 $x$，$y$ 和 $z$ 坐标系的单位柱面。可以用 surf 或 mesh 函数绘制，或不提供输出变量直接画图。
- [X,Y,Z] = cylinder　返回半径为 1 的柱面的 $x$，$y$ 和 $z$ 坐标。
- [X,Y,Z] = cylinder(r)　返回用 $r$ 定义周长曲线的柱面的三维坐标。cylinder 函数将 $r$ 中的每个元素作为半径。
- [X,Y,Z] = cylinder(r,n)　返回用 $r$ 定义周长曲线的柱面的三维坐标。在它周围有 $n$ 个间隔点。
- cylinder(…)　无输出变量，用 surf 函数绘柱面。

下面绘制一个默认柱面，用随机生成的颜色着色。

```
cylinder
axis square
h = findobj('Type','surface');
set(h,'CData',rand(size(get(h,'CData'))))
```

结果如图 11-10 所示。

可以沿纵轴方向改变柱面的半径。例如，

```
t = - pi:pi/10:pi;
[X,Y,Z] = cylinder(2+sin(t));
surf(X,Y,Z)
axis square
```

生成图 11-11。

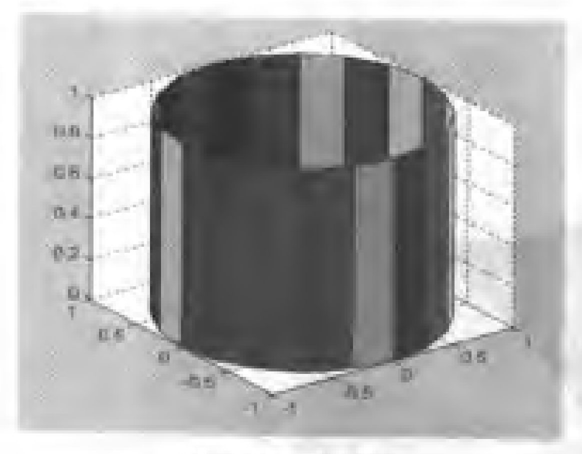


图 11-10　默认柱面

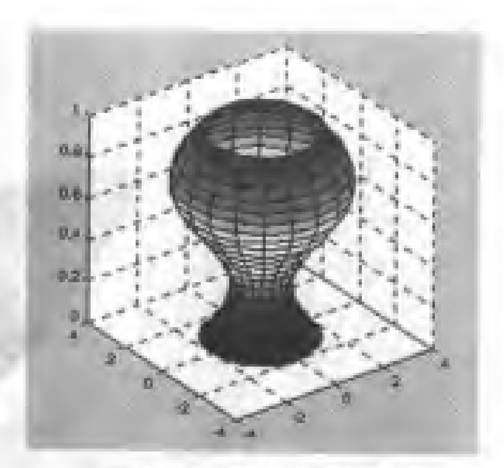


图 11-11　改变柱面的半径

### 2. 球面

绘制曲面的 surf 函数可以有两个额外的向量或矩阵变量，它们用指定的 ***X*** 和 ***Y*** 数据描述曲面。如果 **Z** 是 $m \times n$ 的矩阵，***X*** 是大小为 $n$ 的向量，***Y*** 是大小为 $m$ 的向量，则 mesh(x,y,Z,C)用顶点来描述网格曲面。这些顶点的颜色为 $C(i, j)$，位置为$(x(j), y(i), Z(i, j))$，其中 $x$ 对应于 **Z** 的列，$y$ 对应于 **Z** 的行。

更一般地，如果 ***X,Y,Z*** 和 ***C*** 是维数相同的矩阵，则 mesh(X,Y,Z,C)用顶点来描述网格曲面。这些顶点的颜色为 $C(i, j)$，位置为$(x(i, j), y(i, j), Z(i, j))$。

下面的例子用球坐标绘制一个圆球，并按照 Hadamard 矩阵中的加减模式进行着色，Hadamard 矩阵用于信号处理，是一个正交矩阵。向量 ***theta*** 和 ***phi*** 的取值范围为$-\pi \leqslant \boldsymbol{theta} \leqslant \pi$，$-\pi/2 \leqslant \boldsymbol{phi} \leqslant \pi$。因为 ***theta*** 是一个行向量，***phi*** 是一个列向量，***X, Y*** 和 **Z** 均为向量。

```
k = 5;
n = 2^k-1;
theta = pi*(-n:2:n)/n;
phi = (pi/2)*(-n:2:n)'/n;
X = cos(phi)*cos(theta);
Y = cos(phi)*sin(theta);
Z = sin(phi)*ones(size(theta));
colormap([0 0 0;1 1 1])
C = hadamard(2^k);
surf(X,Y,Z,C)
axis square
```

结果如图 11-12 所示。

除了使用 surf 函数外，MATLAB 还直接提供了绘制球面的 sphere 函数。下面用该函数创建一个默认球面。

```
sphere
axis equal
```

生成图 11-13。

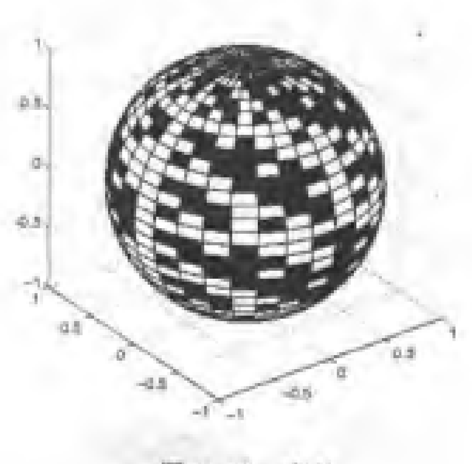



图 11-12　球面

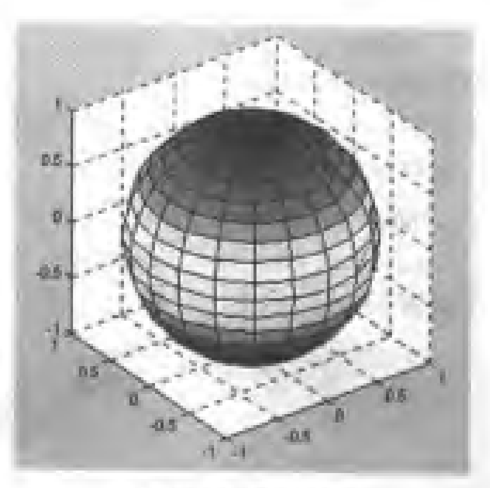



图 11-13　默认球面

### 11.2.3　样条曲面

利用 MATLAB 样条工具箱，可以绘制样条曲面，包括三次样条曲面、B 样条曲面和有理样条曲面等。使用样条工具箱，必须安装该工具箱。关于样条工具箱的使用，可参见文献 8。下面分别演示几种样条曲面，其中用到了样条工具箱的部分函数。

在命令窗口键入下面的命令行，生成一个三次样条曲面。

```
x =.0001+[-4:.2:4]; y = -3:.2:3;
[yy,xx] = meshgrid(y,x); r = pi*sqrt(xx.^2+yy.^2); z = sin(r)./r;
bcs = csapi( {x,y}, z ); fnplt( bcs ), axis([-5 5 -5 5 -.5 1])
```

结果如图 11-14 所示。

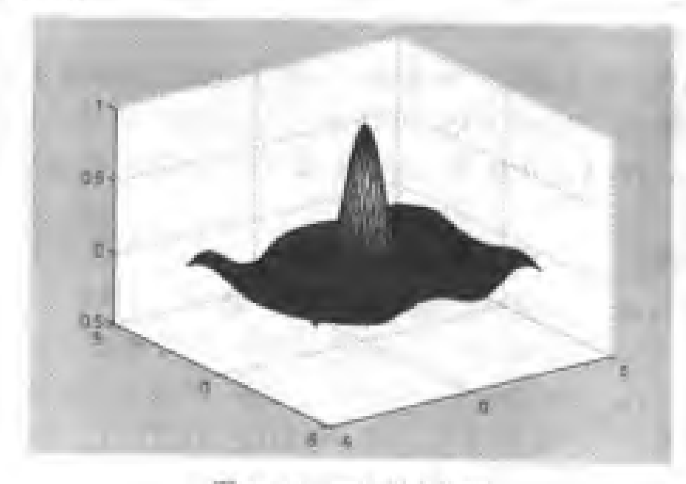



图 11-14　三次样条曲面

键入下面的命令行，生成 B 样条曲面。

```
x = -2:.5:2;
y=-1:.25:1;
[xx, yy] = ndgrid(x,y);
z = exp(- (xx.^2+yy.^2));
sp = spapi({3,4},{x,y},z);
fnplt(sp)
```

结果如图 11-15 所示。

下面的命令行生成一个有理样条曲面，改变 rsmak 函数的 cylinder 参数，可以生成锥面、柱面和球面等多种曲面。

```
fnplt(fncmb(rsmak('cylinder',1,2),[0 0 -1;0 1 0;1 0 0]))
axis equal, axis off, shading interp
```

生成图 11-16，这是一个柱面，进行了插值着色。

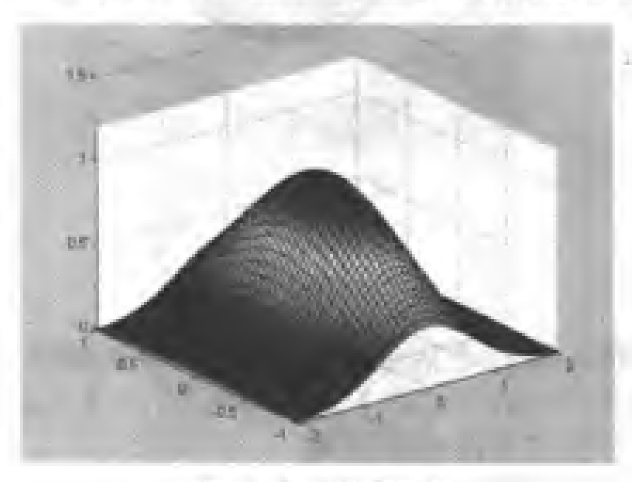


图 11-15　B 样条曲面



图 11-16　有理样条曲面

### 11.2.4　用给定数据绘图

当绘图数据已知时，可以用 mesh，surf 和 surfl 等函数直接绘制网格图、刻面图和曲面图。

**1．网格图**

网格图常用于表现二维平面或三维实体。作为前处理，网格图常常用于建立二维、三维有限元计算的几何模型。其中的网格，可以是三角形网格，也可以是四边形网格，还可以是其他多边形网格。MATLAB 中提供了四边形和三角形两种三维网格图形的生成函数，可以用它们进行绘图。

（1）四边形网格图

用 mesh、meshc 和 meshz 函数绘四边形网格图。调用格式为：

- mesh(X,Y,Z)　绘网格，用 **Z** 确定颜色。颜色与高度成比例。***X*** 和 ***Y*** 为向量，length(X) = n，length(Y) = m,其中[m,n] = size(Z)。本例中，$X(j),Y(i),Z(i,j)$为网格线的交点；***X*** 和 ***Y*** 对应于 **Z** 的列和行。若 ***X*** 和 ***Y*** 为矩阵，则 $X(i,j)$, $Y(i,j)$, $Z(i,j)$为网格线的交点。
- mesh(Z)　用 ***X*** = 1:*n* 和 ***Y*** = 1:*m* 绘一个网格，其中[m,n] = size(Z)。高度 **Z** 为矩形网格

上的单值函数。颜色与高度成比例。

- mesh(...,C)　用矩阵 ***C*** 确定的颜色画网格。若 ***X***,***Y*** 和 ***Z*** 为矩阵，则它们必须与 ***C*** 具有相同的大小。
- meshc(...)　在网格下方画一个等值线图。
- meshz(...)　在网格周围画一个窗帘图。
- h = mesh(...)，h = meshc(...)和 h = meshz(...)　返回网格图对象的句柄。

下面生成 peaks 函数的网格图与等值线图的组合图。

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
colormap colorcube;
axis([-3 3 -3 3 -10 5])
```

生成图 11-17。

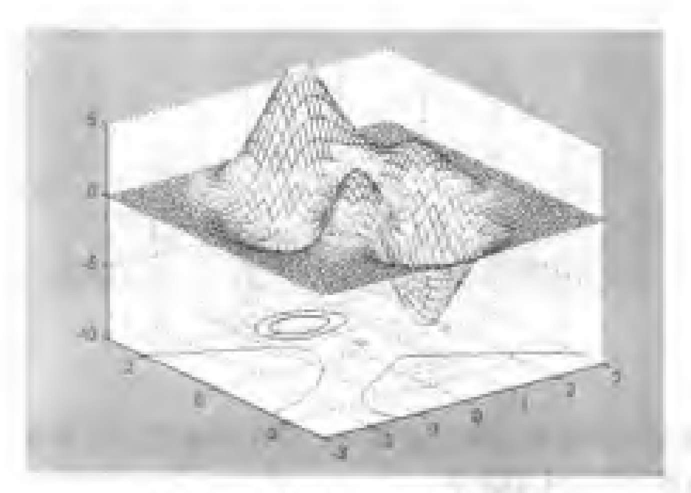


图 11-17　网格图和等值线图的组合图

下面创建 peaks 函数的窗帘图。

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshz(X,Y,Z)
```

结果如图 11-18 所示。

（2）三角形网格图

用 trimesh 函数生成三角形网格图。调用格式为：

- trimesh(Tri,X,Y,Z)　显示由 $m\times3$ 的矩阵 ***Tri*** 定义的三角形网格。***Tri*** 的每一行通过给向量或矩阵 ***X***,***Y*** 和 ***Z*** 赋指数来定义单个三角形。
- trimesh(Tri,X,Y,Z,C)　用同样的方式作为 surf 函数定义 ***C*** 来指定颜色。
- trimesh(...,'PropertyName',PropertyValue...)　为函数创建的阴影图形对象指定附加阴影属性

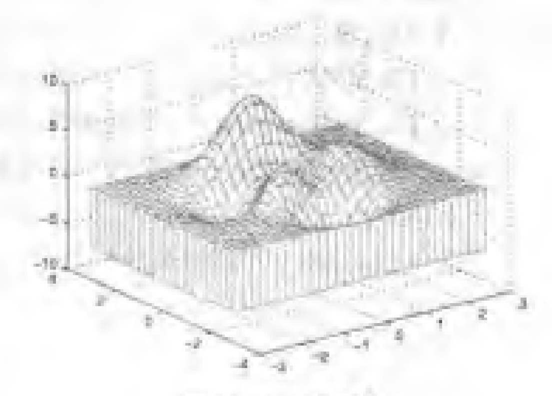


图 11-18　窗帘图

名和属性值。

- h = trimesh(...)　返回阴影图形对象的句柄。

下面创建顶点向量和网格矩阵，然后创建一个三角形网格图。

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trimesh(tri,x,y,z)
```

结果如图 11-19 所示。

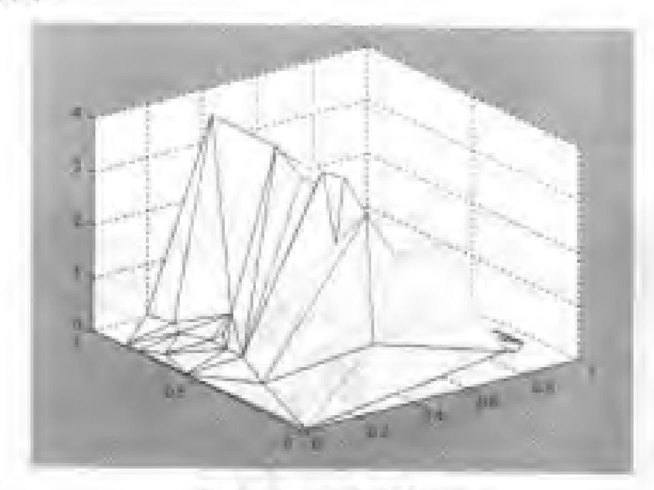


图 11-19　三角形网格图

### 2. 三维刻面图

将三维网格图中的三角形单元或四边形单元用不同颜色进行填充，生成三维刻面图。对应的有三角形刻面图和四边形刻面图。

（1）四边形刻面图

用 surf 函数绘三维刻面图，调用格式为：

- surf(Z)　用 $x = 1:n$ 和 $y = 1:m$，其中[m,n] = size(Z)，根据 **Z** 矩阵中的 $z$ 元素创建一个三维刻面图。高度 **Z** 为一单值函数，定义在一个矩形网格上。**Z** 指定颜色数据和刻面高度，颜色与刻面高度成比例。
- surf(X,Y,Z)　用 **Z** 作为颜色数据和刻面高程数据创建刻面图。***X*** 和 ***Y*** 为向量或矩阵。若 ***X*** 和 ***Y*** 为向量，则 length(X) = n,length(Y) = m，其中[m,n] = size(Z)。
- surf(X,Y,Z,C)　用 ***C*** 定义的颜色创建刻面图。
- surf(...,'PropertyName',PropertyValue)　同时指定刻面属性。
- surfc(...)　在刻面图下方绘一个等值线图。
- h = surf(...) 和 h = surfc(...)　返回刻面图形对象的句柄。

下面生成 peaks 函数的刻面和等值线图。

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z)
colormap colorcube
axis([-3 3 -3 3 -10 5])
```

生成图 11-20。

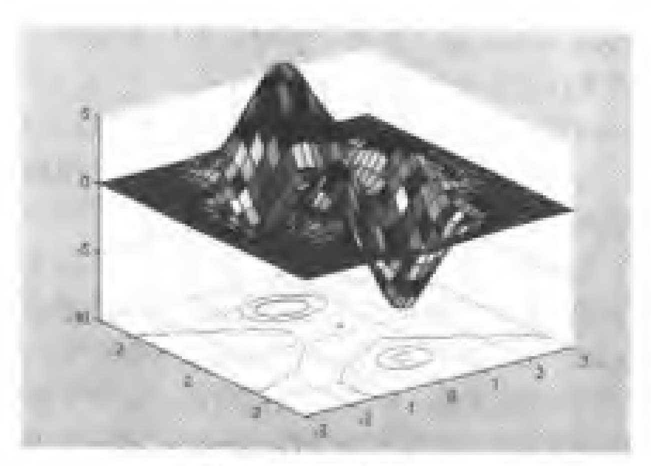


图 11-20　三维刻面图

（2）三角形刻面图

用 trisurf 函数生成三角形刻面图。调用格式为：

- trisurf(Tri,X,Y,Z)　显示 $m \times 3$ 的矩阵 ***Tri*** 定义的三角形网格，并作为刻面。***Tri*** 的每一行通过给向量或矩阵 ***X***,***Y*** 和 **Z** 赋索引值来定义单个三角形。
- trisurf(Tri,X,Y,Z,C)　用同样的方式作为 surf 函数定义 ***C*** 来指定颜色。
- trisurf(...'PropertyName',PropertyValue...)　为函数创建的阴影图形对象指定附加阴影属性名和属性值。
- h = trisurf(...)　返回一个阴影句柄。

下面创建顶点向量和刻面矩阵，然后创建一个三角形刻面图。

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trisurf(tri,x,y,z)
```

生成图 11-21。

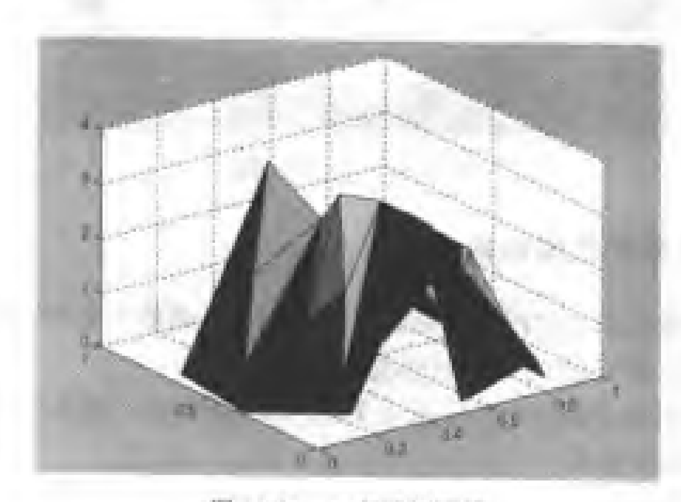


图 11-21　三角形刻面图

### 3. 三维曲面图

三维曲面图在三维刻面图的基础上生成，它对三维刻面图中三角形单元或四边形单元的颜色进行了平滑，因而更接近实体外观。

使用 surfl 函数，用基于颜色查找表的光照绘曲面图。调用格式为：

- surfl(Z)和 surfl(X,Y,Z)　用默认的光源方向设置和默认的阴影模型光线系数创建三维刻面图。**X**,**Y** 和 **Z** 为向量或矩阵，定义曲面的 ***x***, ***y*** 和 ***z*** 分量。
- surfl(...,'light')　用光线对象生成一个彩色的、有明暗色调的曲面。
- surfl(...,s)　指定光源的方向。*s* 是一个二元素或三元素向量，指定曲面到光源的方向 s = [sx sy sz]或 s = [azimuth elevation]。默认设置为从当前视角逆时针旋转 45°。
- surfl(X,Y,Z,s,k)　指定反射常数。***k*** 为四元素向量，定义环境光、漫反射、镜面反射和镜面照射系数等对光照效果的贡献。k = [ka kd ks shine]，默认值为[.55,.6,.4,10]。
- h = surfl(...)　返回曲面图形对象的句柄。

下面用基于颜色查找表的光照生成 peaks 函数的曲面图，如图 11-22 所示。

```
[x,y] = meshgrid(-3:1/5:3);
z = peaks(x,y);
surfl(x,y,z);
shading interp
colormap(colorcube);
axis([-3  3  -3  3  -8  8])
```

生成图 11-22。

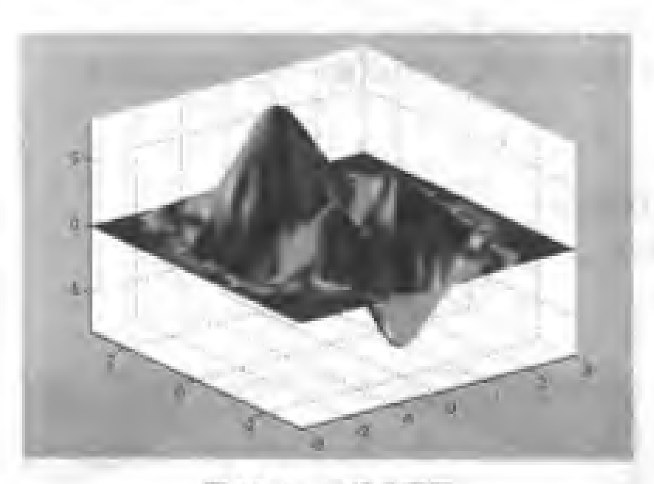


图 11-22　三维曲面图

## 11.2.5　非均匀采样数据的曲面图

生成非均匀采样数据的曲面图，首先需要用 griddata 函数在非均匀间隔点上进行插值，然后根据插值得到的数据用 mesh 和 surf 函数进行绘图。

下面的例子计算指定范围内一些随机点上的 sinc 函数值，然后生成非均匀采样数据，显示为曲面图。包括以下步骤：

- 用 linspace 函数生成非均匀采样数据中一定范围内的均匀间隔值。
- 用 meshgrid 函数，利用 linspace 函数的输出数据生成绘图网格。

- 用 griddata 函数将非均匀采样数据插值到 meshgrid 函数返回的均匀间隔网格中。
- 用绘图函数显示数据。

（1）首先，生成[-8,8]范围内的非均匀采样数据，并用它计算函数。

```
x = rand(100,1)*16 - 8;
y = rand(100,1)*16 - 8;
r = sqrt(x.^2 + y.^2) + eps;
z = sin(r)./r;
```

（2）用 linspace 函数生成一定范围内均匀间隔的随机数。

```
xlin = linspace(min(x),max(x),33);
ylin = linspace(min(y),max(y),33);
```

（3）用这些点生成均匀间隔的网格。

```
[X,Y] = meshgrid(xlin,ylin);
```

（4）用 griddata 函数，根据原始数据点上的函数值，通过插值法计算均匀间隔点上的函数值。下面的语句采用 3 次插值生成新数据。

```
Z = griddata(x,y,z,X,Y,'cubic');
```

（5）然后根据插值后的数据绘网格图，用红色圆点表示非均匀采样点。

```
mesh(X,Y,Z)
axis tight; hold on
plot3(x,y,z,'.','MarkerSize',15)
```

结果如图 11-23 所示。

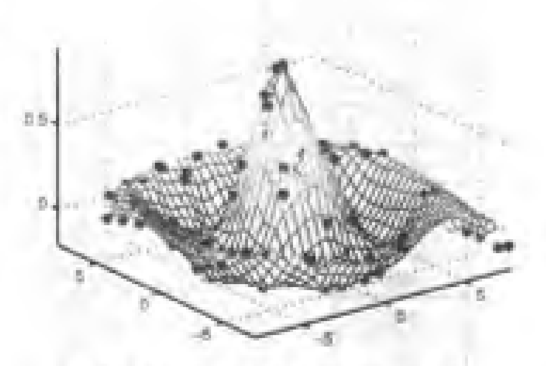


图 11-23　非均匀采样数据的曲面图

## 11.2.6　表面图绘制的数据格式问题

在写作本书之前，作者曾经写过 MATLAB 其他方面的书。读者来信显示，很多读者绘制表面图时最大的困惑不在于如何使用绘图函数，而在于如何组织数据。

MATLAB 绘制三维表面图的数据格式与传统的 Surfer 等的不同，它是矩阵格式的。下面结合一个具体的实例进行介绍。假设有表 11-1 所示的数据，现在利用 *x*、*t* 和 *u* 变量的数据绘三维表面图（此数据由庄陈坚提供）。

表 11-1　绘图原始数据

| x | t | u | x | t | u |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 3 | 0 |
| pi/10 | 0 | 0.3090 | pi/10 | 3 | 0.0154 |
| pi/5 | 0 | 0.5878 | pi/5 | 3 | 0.0293 |
| 3pi/10 | 0 | 0.8090 | 3pi/10 | 3 | 0.0403 |
| 2pi/5 | 0 | 0.9511 | 2pi/5 | 3 | 0.0474 |
| pi/2 | 0 | 1 | pi/2 | 3 | 0.0498 |
| 3pi/5 | 0 | 0.9511 | 3pi/5 | 3 | 0.0474 |
| 7pi/10 | 0 | 0.8090 | 7pi/10 | 3 | 0.0403 |
| 4pi/5 | 0 | 0.5878 | 4pi/5 | 3 | 0.0293 |

续表

| x | t | u | x | t | u |
|---|---|---|---|---|---|
| 9pi/10 | 0 | 0.3090 | 9pi/10 | 3 | 0.0154 |
| pi | 0 | 0 | pi | 3 | 0 |
| 0 | 1 | 0 | 0 | 4 | 0 |
| pi/10 | 1 | 0.1137 | pi/10 | 4 | 0.0057 |
| pi/5 | 1 | 0.2162 | pi/5 | 4 | 0.0108 |
| 3pi/10 | 1 | 0.2976 | 3pi/10 | 4 | 0.0148 |
| 2pi/5 | 1 | 0.3499 | 2pi/5 | 4 | 0.0174 |
| pi/2 | 1 | 0.3699 | pi/2 | 4 | 0.0183 |
| 3pi/5 | 1 | 0.3499 | 3pi/5 | 4 | 0.0174 |
| 7pi/10 | 1 | 0.2976 | 7pi/10 | 4 | 0.0148 |
| 4pi/5 | 1 | 0.2162 | 4pi/5 | 4 | 0.0108 |
| 9pi/10 | 1 | 0.1137 | 9pi/10 | 4 | 0.0057 |
| pi | 1 | 0 | pi | 4 | 0 |
| 0 | 2 | 0 | 0 | 5 | 0 |
| pi/10 | 2 | 0.0418 | pi/10 | 5 | 0.0021 |
| pi/5 | 2 | 0.0795 | pi/5 | 5 | 0.0040 |
| 3pi/10 | 2 | 0.1095 | 3pi/10 | 5 | 0.0055 |
| 2pi/5 | 2 | 0.1287 | 2pi/5 | 5 | 0.0064 |
| pi/2 | 2 | 0.1353 | pi/2 | 5 | 0.0067 |
| 3pi/5 | 2 | 0.1287 | 3pi/5 | 5 | 0.0064 |
| 7pi/10 | 2 | 0.1095 | 7pi/10 | 5 | 0.0055 |
| 4pi/5 | 2 | 0.0795 | 4pi/5 | 5 | 0.0040 |
| 9pi/10 | 2 | 0.0418 | 9pi/10 | 5 | 0.0021 |
| pi | 2 | 0 | pi | 5 | 0 |

表 11-1 中的数据是典型的 Surfer 格式数据，需要单独指定每个点的位置，即横坐标和纵坐标。为了用 MATLAB 绘图，需要首先将该数据转换为 MATLAB 接受的格式。将数据转换为表 11-2 所示的格式。其中，第一行指定 $t$ 变量的取值，第一列指定 $x$ 变量的取值。中间数据为 $u$ 变量的值。每个 $u$ 值的平面位置由其所在的行和列的值确定，高度上的位置由自己的大小确定。

**表 11-2　转换数据格式**

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pi/10 | 0.3090 | 0.1137 | 0.0418 | 0.0154 | 0.0057 | 0.0021 |
| pi/5 | 0.5878 | 0.2162 | 0.0795 | 0.0293 | 0.0108 | 0.0040 |
| 3pi/10 | 0.8090 | 0.2976 | 0.1095 | 0.0403 | 0.0148 | 0.0055 |
| 2pi/5 | 0.9511 | 0.3499 | 0.1287 | 0.0474 | 0.0174 | 0.0064 |
| pi/2 | 1 | 0.3699 | 0.1353 | 0.0498 | 0.0183 | 0.0067 |
| 3pi/5 | 0.9511 | 0.3499 | 0.1287 | 0.0474 | 0.0174 | 0.0064 |
| 7pi/10 | 0.8090 | 0.2976 | 0.1095 | 0.0403 | 0.0148 | 0.0055 |
| 4pi/5 | 0.5878 | 0.2162 | 0.0795 | 0.0293 | 0.0108 | 0.0040 |
| 9pi/10 | 0.3090 | 0.1137 | 0.0418 | 0.0154 | 0.0057 | 0.0021 |
| pi | 0 | 0 | 0 | 0 | 0 | 0 |

得到表 11-2 所示的数据以后，如果只需要查看表面图的基本形态特征，对坐标没有严格要求，可以直接利用 $u$ 变量的数据进行绘图，在命令行键入

```
u=[0        0        0        0        0        0;
   0.3090   0.1137   0.0418   0.0154   0.0057   0.0021;
   0.5878   0.2162   0.0795   0.0293   0.0108   0.0040;
   0.8090   0.2976   0.1095   0.0403   0.0148   0.0055;
   0.9511   0.3499   0.1287   0.0474   0.0174   0.0064;
   1        0.3699   0.1353   0.0498   0.0183   0.0067;
   0.9511   0.3499   0.1287   0.0474   0.0174   0.0064;
   0.8090   0.2976   0.1095   0.0403   0.0148   0.0055;
   0.5878   0.2162   0.0795   0.0293   0.0108   0.0040;
   0.3090   0.1137   0.0418   0.0154   0.0057   0.0021;
   0        0        0        0        0        0];
surf(u)
```

生成图 11-24。图中的平面坐标与所要求的不吻合，但图形的基本形态已经很清晰了。

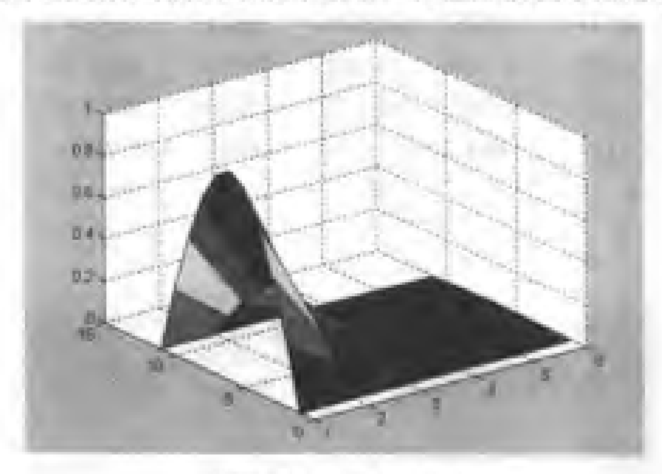


图 11-24　生成表面图

如果要求对平面坐标也有严格控制，可作如下编程。

```
x=[0        0        0        0        0        0;
   0.3142   0.3142   0.3142   0.3142   0.3142   0.3142;
   0.6284   0.6284   0.6284   0.6284   0.6284   0.6284;
   0.9426   0.9426   0.9426   0.9426   0.9426   0.9426;
   1.2568   1.2568   1.2568   1.2568   1.2568   1.2568;
   1.5710   1.5710   1.5710   1.5710   1.5710   1.5710;
   1.8852   1.8852   1.8852   1.8852   1.8852   1.8852;
   2.1994   2.1994   2.1994   2.1994   2.1994   2.1994;
   2.5136   2.5236   2.5236   2.5236   2.5236   2.5236;
   2.8278   2.8378   2.8378   2.8378   2.8378   2.8378;
   3.1420   3.1420   3.1420   3.1420   3.1420   3.1420];
t=[0        1        2        3        4        5;
   0        1        2        3        4        5;
```

```
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5;
  0      1      2      3      4      5];
u=[0      0      0      0      0      0;
  0.3090  0.1137  0.0418  0.0154  0.0057  0.0021;
  0.5878  0.2162  0.0795  0.0293  0.0108  0.0040;
  0.8090  0.2976  0.1095  0.0403  0.0148  0.0055;
  0.9511  0.3499  0.1287  0.0474  0.0174  0.0064;
  1       0.3699  0.1353  0.0498  0.0183  0.0067;
  0.9511  0.3499  0.1287  0.0474  0.0174  0.0064;
  0.8090  0.2976  0.1095  0.0403  0.0148  0.0055;
  0.5878  0.2162  0.0795  0.0293  0.0108  0.0040;
  0.3090  0.1137  0.0418  0.0154  0.0057  0.0021;
  0       0       0       0       0       0];
surf(x,t,u)
```

生成图 11-25。

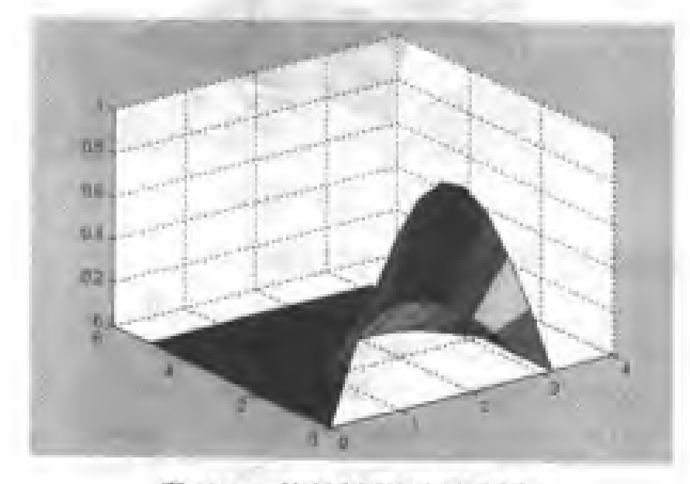
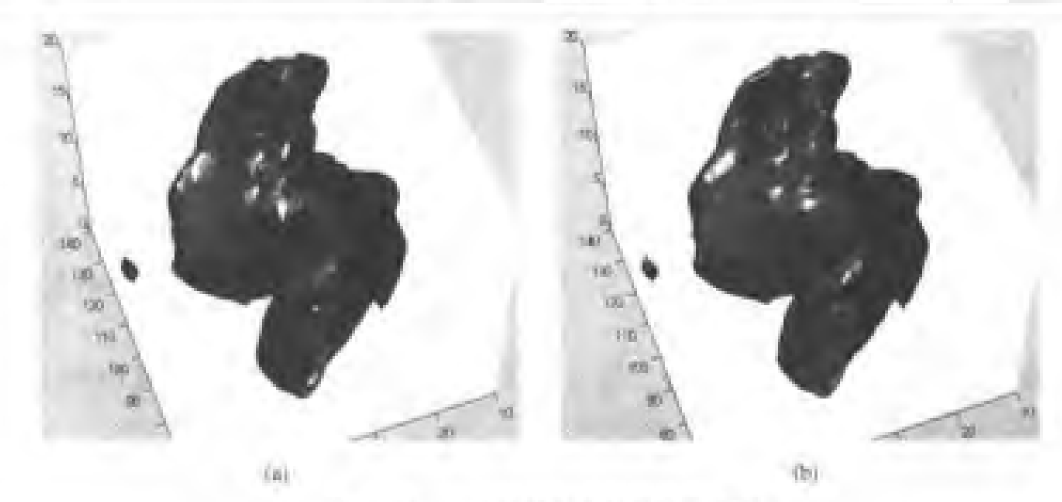


图 11-25　控制表面图的平面坐标

## 11.3　多边形对象模型

由一个或多个相连或不相连的多边形组成的对象称为面片图形对象，也常称为多边形对象模型。面片对于真实世界中对象如飞机、汽车的模型建立和任意形状二、三维对象的绘制很有用。图 11-26 是一个较复杂对象的多边形对象模型，其中图（a）和图（b）分别为平滑处理前、后的显示效果。



(a)　　　　(b)

图 11-26　一个比较复杂的多边形模型

创建多边形对象，需要定义面片。只需要指定顶点坐标和某种形式的颜色数据就可以定义面片了。面片支持多种着色方式，这一点对于用几何形体进行数据可视化很有用。

定义面片主要有两种方式：

- 指定每个多边形的顶点坐标，MATLAB 按顺序连接这些顶点，形成面片。
- 指定每个顶点的坐标值和一个指定如何把这些点连成面的矩阵。

第 2 种方式对于有多个小面的面片更合适，因为这种方式定义面片时需要的数据更少，多个小面共享的顶点只需要定义一次就够了。后面会对两种方式都举例进行介绍。

有好几个 MATLAB 函数都可以创建面片对象，如 fill,fill3,isosurface,isocaps,有些 contour 函数和 patch 函数等。下面集中结合 patch 函数进行介绍。

### 11.3.1　patch 函数

有两种形式的 patch 函数，一种是高级语法形式，另一种是低级语法形式。使用的语法形式不同，patch 函数的表现也不同。

使用高级语法形式时，MATLAB 根据指定的颜色值自动确定如何对每个小面进行着色。只要将 x, y, z 坐标值和颜色数据按照正确的顺序选作 patch 函数的变量，使用高级语法可以不用管它们的属性名。变量的正确顺序为

```
patch(x,y,z,colordata)
```

但是，必须指定颜色数据，这样 MATLAB 才能确定使用哪种颜色。如果不指定颜色数据，则 MATLAB 返回下面的错误。

```
patch(sin(t),cos(t))
??? Error using ==> patch
Not enough input arguments.
```

低级语法格式只接受属性名/属性值配对变量，而且不会自动对小面进行着色，除非还改变了 FaceColor 属性的值。例如，下面的语句绘一个小面颜色为白色的面片。

```
patch('XData',sin(t),'YData',cos(t))
```

因为 FaceColor 属性的默认值指定白色，即

```
get(0,'FactoryPatchFaceColor')
```

```
ans =
1     1     1
```

### 11.3.2 用 patch 函数创建面片

使用 patch 函数有两种方法生成面片，一种是通过指定每个小面的顶点坐标和颜色来创建，另一种是指定所有顶点的坐标和各小面的顶点和颜色来创建。下面用这两种方法创建 1 个有 3 个小面的扇面。

首先用小面坐标和颜色进行创建。用 x, y 和 z 向量定义小面的顶点，用 tcolor 向量定义小面的颜色。在命令窗口键入下面的命令行：

```
x = [0 0 0;1 .5 0;1 1 .5];
y = [0 0 0;.5 1 1;0 .5 1];
z = [.5 .5 .5;1 1 1;1 1 1];
tcolor(1,1,1:3) = [.7 .7 .7];
tcolor(1,2,1:3) = [1 1 1];
tcolor(1,3,1:3)=[.3 .3 .3];
patch(x,y,z,tcolor);
axis([0 1 0 1 0 2]);
view(3)
```

生成图 11-27，正好是 3 个小三角形组成的扇面。

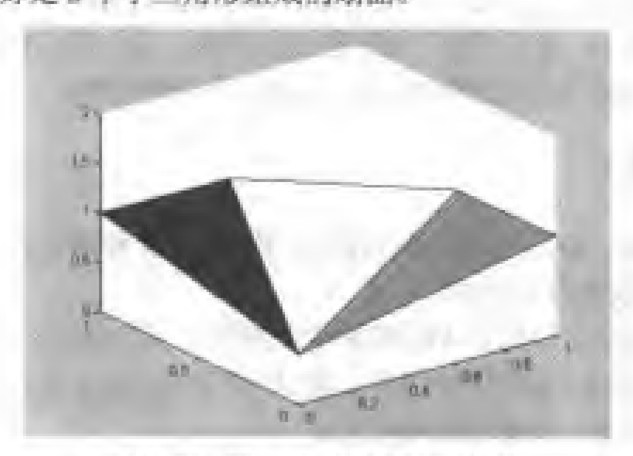


图 11-27　用 patch 函数创建一个扇面

下面用第 2 种方法进行创建。用 vert 向量定义各顶点，用 fac 向量定义小面。键入下面的代码：

```
vert = [0 0 .5;1 0 1;1 .5 1;.5 1 1;0 1 1];
fac = [1 3 2;1 4 3;1 5 4];
tcolor = [.7 .7 .7;1 1 1;.3 .3 .3];
patch('Faces',fac,'Vertices',vert,'FaceVertexCData',tcolor,...
      'FaceColor','flat');
axis([0 1 0 1 0 2]);
view(3)
```

生成与图 11-27 相同的结果。

读者可以从内存和遍历两个方面比较这两种方法的优劣。面片的着色是很重要的，将在下一章进行介绍。

## 11.4　消隐控制

从某个视点观察三维图形时，总会有部分线或面被别的对象或它自身挡住。绘制三维图形时不显示这部分线或面有利于体现图形的三维特征，否则会使图形看起来杂乱无章。绘制三维图形时使被遮挡的图形部分不显示出来的技术通常称为消隐，包括线的消隐和面的消隐。消隐是三维图形绘制的一项关键技术，也是难点。

默认时，MATLAB 绘制三维图形时会进行消隐处理。利用下面的命令，可以取消消隐：

```
hidden off
```

图 11-28 是取消消隐以后绘制的一幅网格图。

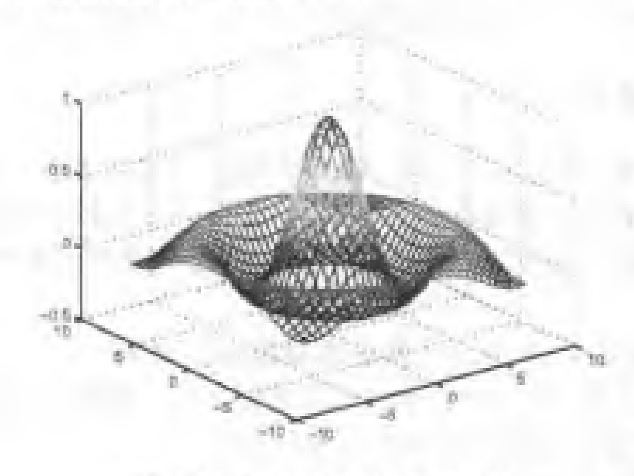


图 11-28　取消消隐以后的网格图

# 第 12 章　三维模型的着色

建立模型以后，需要对模型进行着色。着色使模型更美观，立体感更强。采用插值着色，可以使模型更显真实。另外，在三维模型上用颜色也可以表达一定的信息，这就是所谓的第四维。与建模一样，分表面模型和多边形模型两种情况进行讨论。

## 12.1　网格图、刻面图和曲面图的着色

可以通过给网格图和表面图着色来增强它们的表现力。MATLAB 可以将特定数据值映射给指定的颜色，或者可以将整个范围内的数据映射给名为颜色查找表的预定义范围内的颜色。

### 12.1.1　主要的着色技术

MATLAB 主要采用了 3 种着色技术，即索引着色、真彩色着色和纹理映射。

- 索引着色　MATLAB 通过给每个数据点一个索引值来建立与图形颜色查找表之间的映射关系。MATLAB 应用这些颜色的方式与着色类型即采用平面着色、刻面着色还是插值着色有关。
- 真彩色着色　MATLAB 用分别指定的颜色（即 RGB 分量）对表面图进行着色。MATLAB 应用这些颜色的方式与着色类型即是平面着色、刻面着色还是插值着色有关。要想精确地进行渲染，需要计算机具备 24 位的显示能力。
- 纹理映射　纹理映射将一幅二维图像映射到三维表面上。

颜色数据的类型确定 MATLAB 选择什么样的技术进行着色。创建表面图时，如果没有提供颜色数据，MATLAB 根据 $z$ 值生成颜色查找表的索引值；如果指定一个颜色数据数组，其大小与 $z$ 数据的相同，使用索引着色；如果指定一个 $m\times n\times 3$ 的颜色数据数组，为 $m\times n$ 数据数组中的每个元素定义一个 RGB 颜色分量，使用真彩色着色。

### 12.1.2　颜色查找表

每个 MATLAB 图形窗口都有一个颜色查找表与之相连。颜色查找表是一个 3 列矩阵，其长度等于它定义的颜色数目。矩阵的每一行指定 3 个 0 和 1 之间的值，定义一种颜色。这些值定义 RGB 分量，即红色、绿色和蓝色组分的强度。

使用没有变量的 colormap 函数可以返回当前图形的颜色查找表。例如，MATLAB 默认的颜色查找表包括 64 种颜色，第 57 种是红色。

```
cm = colormap;
cm(57,:)
ans =
   1 0 0
```

表 12-1 中列出了一些颜色的 RGB 定义。

**表 12-1　一些颜色的 RGB 定义**

| 红 | 绿 | 蓝 | 颜 色 |
|---|---|---|---|
| 0 | 0 | 0 | 黑色 |
| 1 | 1 | 1 | 白色 |
| 1 | 0 | 0 | 红色 |
| 0 | 1 | 0 | 绿色 |
| 0 | 0 | 1 | 蓝色 |
| 1 | 1 | 0 | 黄色 |
| 1 | 0 | 1 | 粉红 |
| 0 | 1 | 1 | 青色 |
| 0.5 | 0.5 | 0.5 | 灰色 |
| 0.5 | 0 | 0 | 深红色 |
| 1 | 0.62 | 0.40 | 紫铜色 |
| 0.49 | 1 | 0.83 | 碧绿色 |

可以通过 MATLAB 数组操作或生成任何有用映射，包括 hsv,hot,cool,summer 和 gray 等的函数创建颜色查找表。每个函数有一个可选参数，它指定生成的颜色查找表的行数。例如，

```
hot(m)
```

创建一个 $m \times 3$ 的矩阵，它的行指定一个颜色查找表所表示的颜色的 RGB 分量。这个颜色查找表所表示的颜色从黑色变为红色、橘红、黄色，直至白色。

如果不指定颜色查找表的长度，MATLAB 会创建与当前颜色查找表长度相同的颜色查找表。如果在每个图形窗口中使用长颜色查找表（>64 色），对于操作系统来说，当激活焦点在窗口内移动时在不同颜色查找表中进行交换是必要的。

MATLAB 支持下面一些颜色查找表函数：

- autumn　从红色向橘黄色、黄色平稳过渡。
- bone　为含有较高的蓝色组分的 gray 颜色查找表。
- colorcube　该函数包含 RGB 颜色空间中尽可能多的规则间隔的颜色，它试图提供更多的灰色、纯红、纯绿和纯蓝。
- cool　由青色和洋红阴影组成的颜色。它在青色和洋红之间平滑过渡。
- copper　在黑色和亮铜色之间平滑过渡。
- flag　由红色、白色、蓝色和黑色组成。每次索引值增加时，该颜色查找表会完全改变颜色。
- gray　返回线性灰阶颜色查找表。
- hot　在黑色、红色、橘红色、黄色和白色之间平滑过渡。
- hsv　变化 HSV 颜色模型中的色度组分。颜色从红色开始，然后为黄色、绿色、青色、蓝色、洋红，最后返回到红色。该颜色查找表特别适合于显示周期性函数。hsv(m)与 hsv2rgb([h ones(m,2)])相同，其中 $h$ 为线性坡度，h = (0:m−1)'/m。
- jet　在蓝色、青色、黄色、橘红色、红色之间过渡。由 hsv 颜色查找表变化而来。
- lines　生成颜色由坐标系对象的 ColorOrder 属性和灰色阴影确定的颜色查找表。

- pink　包含品红色的柔和阴影。该颜色查找表使得可以对灰度照片进行棕褐色化。
- prism　重复红色、橘红色、黄色、绿色、蓝色和紫色等 6 种颜色。
- spring　由洋红和黄色阴影组成。
- summer　由绿色和黄色阴影组成。
- white　白色色图。
- winter　由蓝色和绿色阴影组成。

colorbar 函数在图形窗口中显示当前的颜色查找表所表示的颜色，它或者垂直或者水平地显示在图形旁边。例如，下面的语句生成一个表面图和一个垂直的颜色条，该条形图对应于当前的颜色查找表。注意颜色条是如何用坐标轴标签值（作为索引值）表现数据值与颜色之间的映射关系的。

```
[x,y] = meshgrid([-2:.2:2]);
Z = x.*exp(-x.^2-y.^2);
surf(x,y,Z,gradient(Z))
colorbar
```

生成的图形如图 12-1 所示。

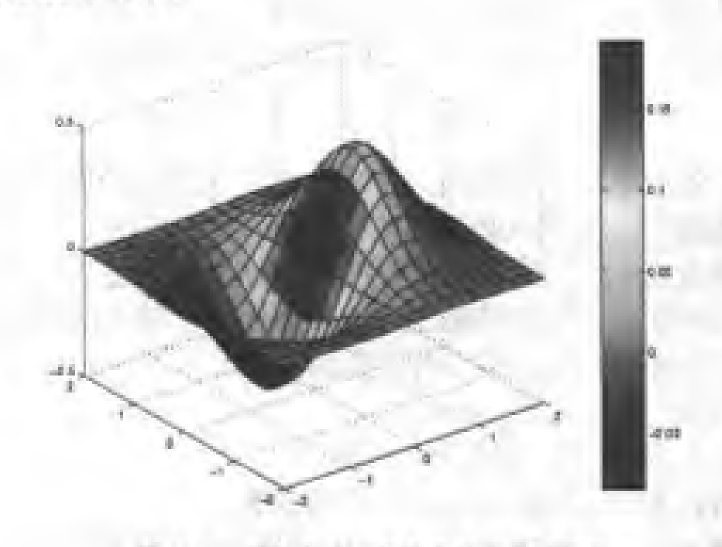


图 12-1　用色条表示颜色查找表中的颜色

为了方便地设置图形的颜色查找表，MATLAB 提供了一个颜色查找表编辑器。用 colormapeditor 命令启动颜色查找表编辑器，并在编辑器中显示当前图形窗口的颜色查找表。如下所示，在命令窗口键入

```
colormapeditor
```

打开“Colormap Editor”窗口，如图 12-2 所示。

编辑器中，当前图形窗口颜色查找表中的颜色用一系列颜色渐变的矩形单元表示。矩形单元下方有几个节点指针，移动节点指针，可以改变颜色查找表中的颜色范围，并改变当前图形的显示。“Current color info”方框中显示颜色的索引值和颜色数据。在“Interpolating colorspace”下拉式列表框中进行选择，可以转换颜色空间。这里提供了 RGB 和 HSV 两种颜色空间。

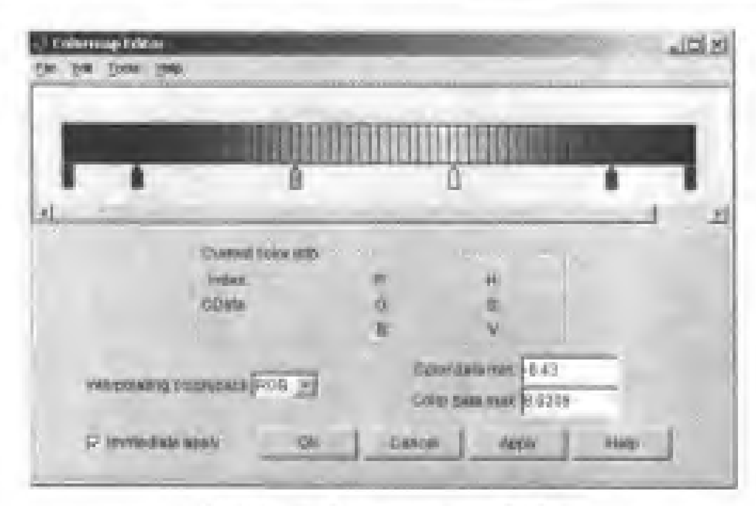


图 12-2　"Colormap Editor" 窗口

选择和移动节点指针，可以改变颜色查找表中颜色的范围。编辑器会在两个节点指针之间进行颜色插值。双击节点指针，并在弹出的颜色选择器中进行选择，可以改变节点单元的颜色。改变节点颜色以后，根据相邻节点的颜色重新进行插值计算。针对节点指针有多种操作，如表 12-2 所示。

**表 12-2　针对节点的操作说明**

| 操　作 | 说　明 |
| --- | --- |
| 添加一个节点 | 在对应的节点单元下方单击 |
| 选择一个节点 | 单击节点 |
| 选择多个节点 | 对于相邻节点，单击第一个节点，然后按下<Shift>键的同时单击最后一个节点；对于不相邻节点，单击第一个节点，然后按下<Ctrl>键的同时单击后面的节点 |
| 移动一个节点 | 用鼠标选择或拖拉节点，或者选择节点并单击向左或向右键 |
| 移动多个节点 | 选择多个节点并单击向左或向右键 |
| 删除一个节点 | 选择节点，然后单击<Delete>键 |
| 删除多个节点 | 选择节点，然后单击<Delete>键 |
| 显示节点的颜色选择器 | 双击节点 |

图 12-3 是使用默认的颜色查找表得到的曲面图。在命令窗口键入下面的命令行：

```
[x,y,z]=peaks(30);
surf(x,y,z)
```

在颜色查找表编辑器中重新设置颜色，如图 12-4 所示，设置为 cool 颜色查找表的颜色。图 12-3 随即变成图 12-5 所示的颜色。

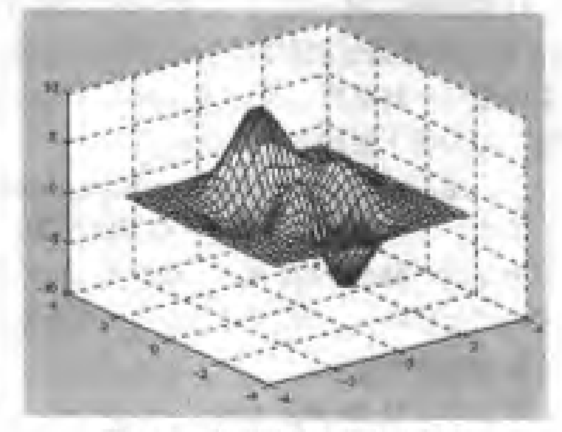


图 12-3　使用默认的颜色查找表

### 12.1.3　索引着色表面——直接映射和比例化映射

MATLAB 可以使用两种不同的方法将索引颜色数据映射到颜色查找表，即直接映射和比例化映射。

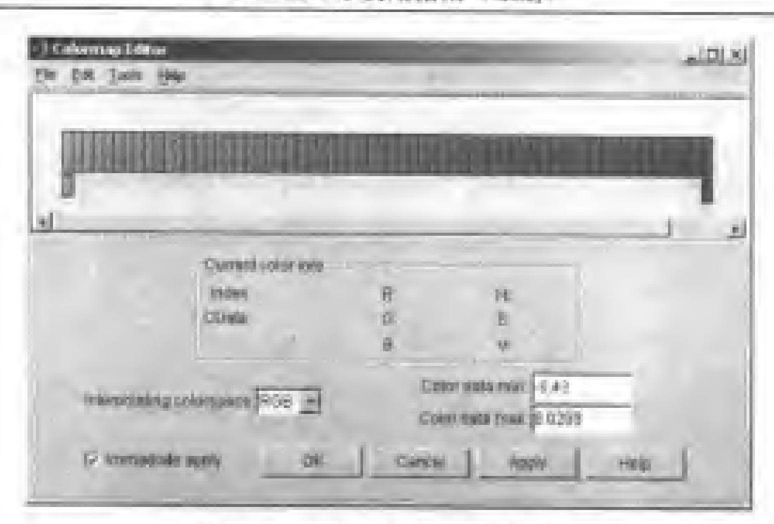


图 12-4　在颜色查找表中调整颜色

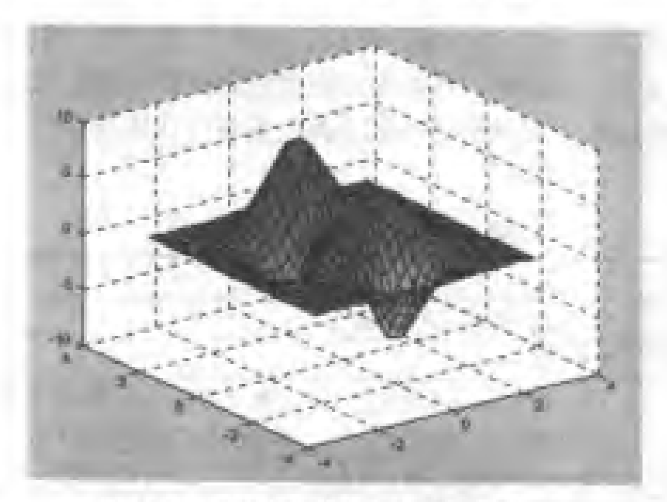


图 12-5　调整颜色查找表以后的图形显示

直接映射是直接将颜色数据作为索引值进行映射。例如，值 1 指向颜色查找表中的第 1 种颜色，值 2 指向第 2 种颜色，依此类推。如果颜色数据不是非整型的，则 MATLAB 将它圆整到 0。大于颜色查找表中的颜色个数的值设置为等于颜色查找表中的最后一种颜色。小于 1 的值设置为 1。

比例化映射用一个 2 元素向量[cmin cmax]控制颜色数据向图形颜色查找表的映射。cmin 指定将数据值映射到颜色查找表中的第一种颜色，cmax 指定将数据值映射到颜色查找表中的最后一种颜色。cmin 和 cmax 之间的数据值线性映射到表中第一种和最后一种之间的颜色。可以用下面的表达式表示：

```
colormap_index = fix((color_data-cmin)/(cmax-cmin)*cm_length)+1
```

其中，cm_length 是颜色查找表的长度。

默认时，MATLAB 设置 cmin 和 cmax 时会包括所有图形对象的颜色数据。但是也可以设置为其他范围。这样，可以在一个图形窗口中显示多个坐标系，以及用颜色查找表的不同部分显示各个坐标系。

默认时，MATLAB 使用比例化映射。使用直接映射，必须在创建图形时取消比例化。例如，

```
surf(Z,C,'CDataMapping','direct')
```

用单个矩阵变量创建表面图时，比如 surf(Z)，其中变量 Z 同时指定高度和表面颜色，MATLAB 会对 Z 进行转换，以获取进入当前颜色查找表的索引值。

也可以使用两个矩阵变量，如下面的语句用第 2 个变量单独指定颜色。

```
surf(Z,C)
```

### 12.1.4 示例——表面曲率向颜色映射

表面图的拉普拉斯算子与它的曲率有关，对于生成类似函数 i^2+j^2 确定的形状的函数，该算子为正；对于生成类似函数-(i^2+j^2)确定的形状的函数，该算子为负。函数 del2 计算任何矩阵的离散拉普拉斯算子。例如，下面用 del2 函数确定 peaks 函数返回的数据的颜色。

```
P = peaks(40);
C = del2(P);
surf(P,C)
colormap hot
```

生成的图形如图 12-6 所示。通过将拉普拉斯算子用于数据来创建颜色数组很有用，因为它会使得具有相似曲率的区域用相同的颜色进行描绘。

试将图 12-2 与下面语句生成的表面图进行比较，它使用了相同的颜色查找表，但是将 z 值相近的区域映射为具有相同的颜色。

```
surf(P)
colormap hot
```

生成图 12-7。

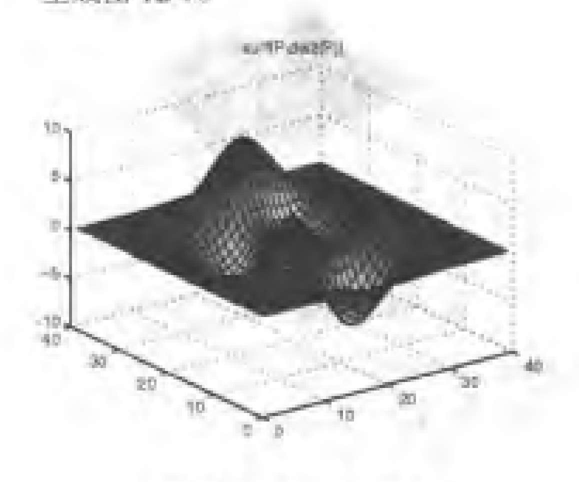



图 12-6 生成的表面图

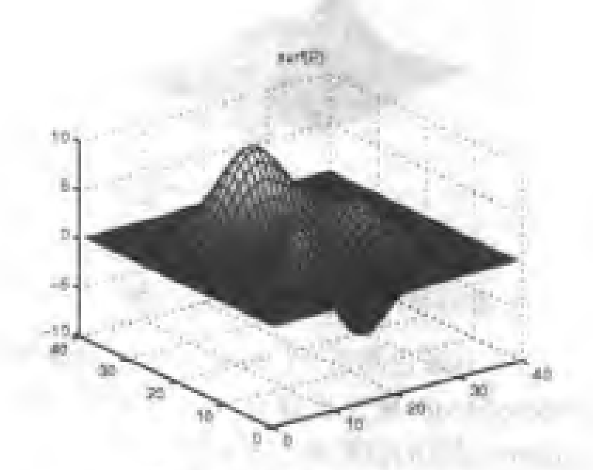


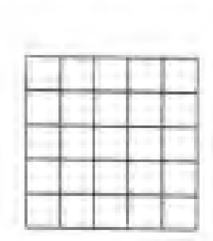

图 12-7 修改颜色后的表面图

### 12.1.5 真彩色表面

24 位显示的计算机系统可以显示超过 1600 万种颜色。有了这种能力，可以将颜色数据直接定义成 RGB 值，而不需要用索引值进行颜色映射。

如图 12-8 所示，用一个 $m \times n \times 3$ 的数组指定真彩色，其中 $z$ 的大小为 $m \times n$。图中左图

为定义表面图的矩阵，右图为定义该图 RGB 值的矩阵。



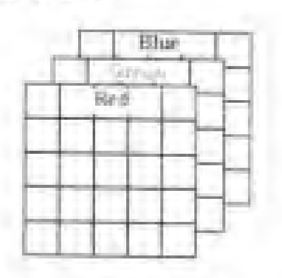



图 12-8　定义表面图及其真彩色的矩阵

下面的语句用 peaks 矩阵数据创建一个随机着色的表面图。

```
Z = peaks(25);
C(:,:,1) = rand(25);
C(:,:,2) = rand(25);
C(:,:,3) = rand(25);
surf(Z,C)
```

效果如图 12-9 所示。

也可以用索引值设置表面属性，例如，

```
surf(Z,C,'FaceColor','interp','FaceLighting','phong')
camlight right
```

绘图效果如图 12-10 所示。

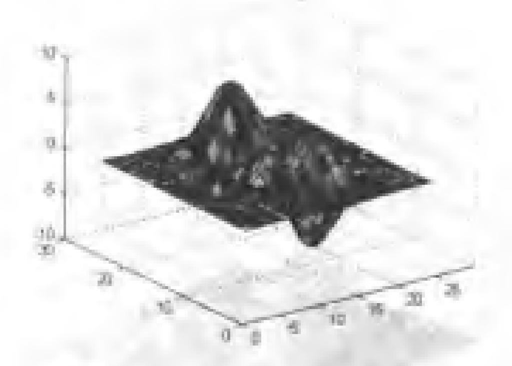


图 12-9　一个随机着色的表面图

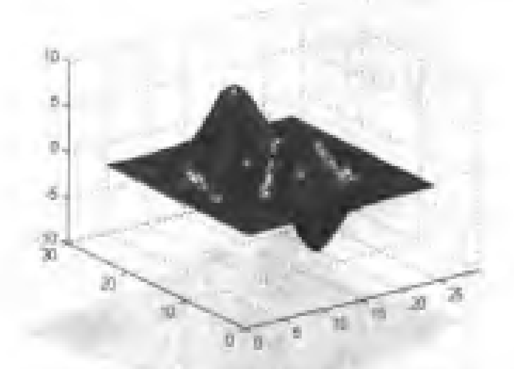


图 12-10　用索引值设置表面属性

显示真彩色图形时，MATLAB 总是使用 OpenGL 或 zbuffer 渲染法。如果 Figure 对象的 RendererMode 属性设置为 auto，则不管什么时候指定真彩色数据，MATLAB 会自动将 Renderer 属性的值转换为 zbuffer。

如果将 Renderer 属性设置为 Painters，并试图用真彩色定义一个图像、面片或表面对象，则 MATLAB 会返回一个警告信息并不对对象进行渲染。

### 12.1.6　纹理映射

纹理映射技术通过变换颜色数据将二维图像映射到三维表面上。它允许颜色数据数组的维数与定义表面图的数据不同。可以将图像映射到任意大小的表面上，MATLAB 会对纹

理颜色数据进行插值，使它可以映射到整个表面上。

下面的例子用 sphere 函数创建一个球形表面，然后将一幅地球图像映射到这个表面上。因为地球图像是从某一侧看到的地球影像，所以这里只将图像映射到球体表面的一侧。

```
load earth
sphere; h = findobj('Type','surface');
hemisphere = [ones(257,125),...
          X,...
          ones(257,125)];
set(h,'CData',flipud(hemisphere),'FaceColor','texture
map')
colormap(map)
axis equal
view([90 0])
set(gca,'CameraViewAngleMode','manual')
view([65 30])
```

纹理映射效果如图 12-11 所示。

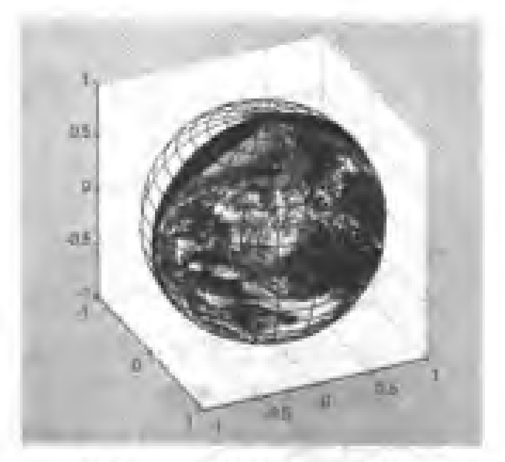


图 12-11　球体的纹理映射效果

## 12.2　多边形模型的着色

面片着色与曲面对象的着色方式不同，主要体现在面片不会根据每个顶点的 z 坐标自动生成颜色数据。用户必须另外对面片进行着色，否则 MATLAB 会用默认颜色进行绘制。给面片着色，通常有 3 种方法：

- 给所有小面着一种颜色；
- 给每个小面着不同的颜色，用于刻面着色；
- 给每个顶点着一种颜色，用于插值着色。

多边形模型中面片的着色有两种方式，即索引着色方式和真彩色着色方式。MATLAB 根据颜色数据的维数确定如何进行解释。如果每个面片、每个小面或每个顶点只指定 1 个数值值，则 MATLAB 将数据解释为索引颜色数据。如果每个面片、小面或顶点都有 3 个数值值，则 MATLAB 将数据解释为 RGB 值。

### 12.2.1　面片只有一个小面的情况

第 3 章中使用 patch 函数的第 1 种语法格式时，MATLAB 将第 3 个（或如果存在 z 坐标，将第 4 个）变量作为颜色数据。如果想用 $x$，$y$ 和 $z$ 坐标定义一个面片，但是没有指定颜色，则 MATLAB 将 $z$ 坐标解释为颜色数据，然后绘制一个二维面片。例如，下面的语句绘制一个面片，它所有顶点的 $z$ 值都为 0，通过利用顶点颜色进行插值来获得小面的颜色。

```
h = patch(sin(t),cos(t),1:length(t))
```

相反，下面的语句也绘制一个面片，$z$ 值增加处的顶点显示为黄色。

```
h = patch(sin(t),cos(t),1:length(t),'y')
```

**1．创建一个单独的多边形**

简单地讲，多边形指的是只有一个小面的面片。要想创建一个多边形，用类似下面形式的语句指定顶点坐标和颜色数据。

```
patch(x-coordinates,y-coordinates,[z-coordinates],colordata)
```

例如，下面这些语句显示一个有 10 条边的多边形，中间用黄色填充，边为黑色。这里使用了 axis equal，横轴和纵轴的度量单位相同，所以这个多边形正确地显示为正多边形。

```
t = 0:pi/5:2*pi;
patch(sin(t),cos(t),'y')
axis equal
```

生成的多边形如图 12-12 所示。

第 1 个和最后 1 个顶点不需要一致，MATLAB 会自动闭合面片的每一个小面。实际上，通常最好只将每个顶点定义一次，特别是使用插值法进行着色时。

**2．用插值法进行着色**

实际上，面片着色的许多方面都是可以控制的。例如，下面不指定一种单色，而是提供很多数值型值，利用它们将每个顶点的颜色映射为图形颜色查找表中的一种颜色。

```
a = t(1:length(t) -1);
patch(sin(a),cos(a),1:length(a),'FaceColor','interp')
colormap cool;
axis equal
```

显示效果如图 12-13 所示。

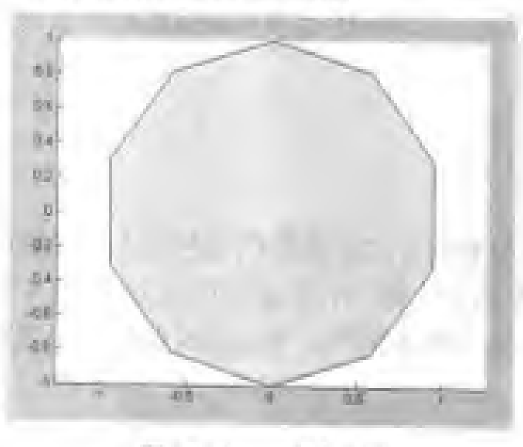


图 12-12　一个多边形

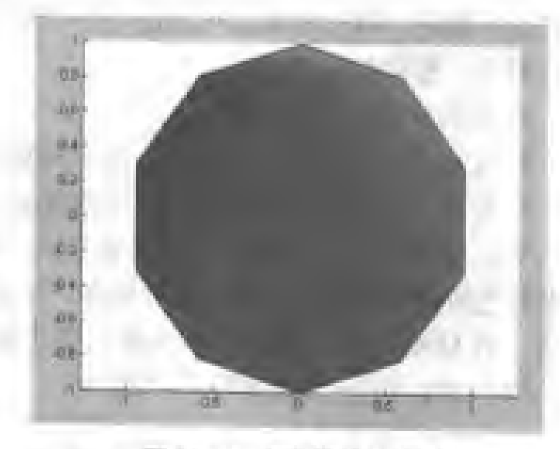


图 12-13　用插值法进行着色

现在，MATLAB 在面片的各个小面上进行插值计算，然后根据插值结果进行着色。可以以相同的方式对面片的边进行着色。命令为

```
patch(sin(t),cos(t),1:length(t),'EdgeColor','interp')
```

## 12.2.2　面片有多个小面的情况

如果将 *x*，*y* 和 *z* 坐标变量指定为向量，则 MATLAB 通过连接这些点来绘制单独的多边形。如果变量是矩阵，则 MATLAB 根据每列的数据绘一个多边形，用多个小面生成一个面片。这些小面不需要彼此连接并且可以是自相交的。

另外，也可以通过指定每个顶点的坐标和组成小面的连接顺序来生成面片。下面的例子演示了这两种技巧。

**1．用 patch 函数创建一个立方体**

第 10 章已经介绍了用 patch 函数创建面片的方法，这里不再重复。下面用顶点/小面方式创建一个白色立方体。

```
vert=[1 1 1;1 2 1;2 2 1;2 1 1;1 1 2;1 2 2;2 2 2;2 1 2];
fac=[1 2 3 4;2 6 7 3;4 3 7 8;1 5 8 4;1 2 6 5;5 6 7 8];
patch('faces',fac,'vertices',vert,'FaceColor','w');
view(3);
axis square;
```

结果如图 12-14 所示。

**2．刻面着色**

刻面着色是给每个小面指定一种颜色以后得到的一种视觉效果。例如，用顶点/小面方式和 FaceVertexCData 属性定义颜色，下面的语句给每个小面指定一种颜色，并将 FaceColor 属性的值设置为 flat。

```
patch('faces',fac,'vertices',vert, 'FaceVertexCData',...
                                    hsv(6),'FaceColor','flat');
```

因为 FaceVertexCData 属性指定的真实颜色与 MATLAB 颜色查找表具有相同的格式，即都是元素为 RGB 值的 $n\times3$ 的数组。本例用颜色查找表 hsv 生成得到刻面颜色所需要的 6 种颜色。

刻面着色效果如图 12-15 所示。

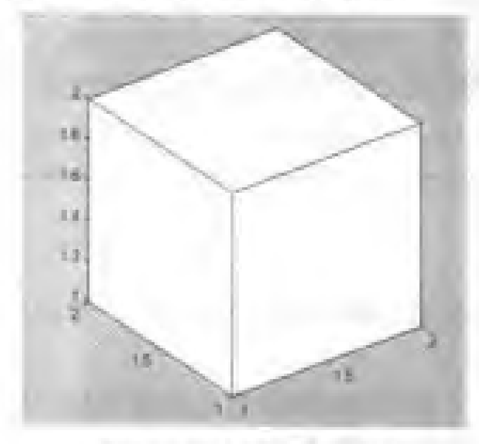


图 12-14　创建一个立方体

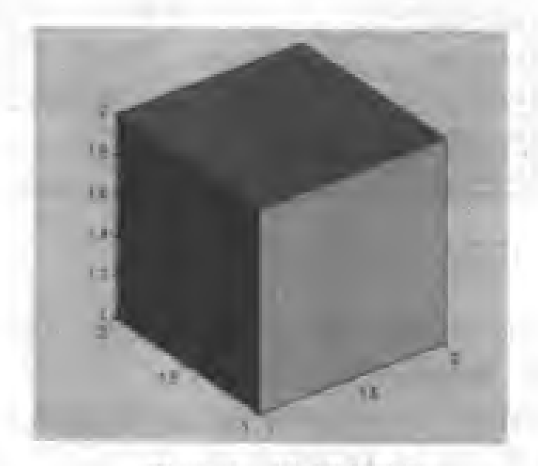


图 12-15　刻面着色效果

**3．插值着色**

插值着色是利用每个小面顶点的颜色进行插值计算，得到小面各边和小面内部各点上的颜色值，然后进行着色。所以，进行插值着色，必须给每个顶点指定一种颜色，并设置 FaceColor 属性的值为 interp，即

```
patch('faces',fac,'vertices',vert, 'FaceVertexCData',...
                                    hsv(8),'FaceColor','interp');
```

生成的效果如图 12-16 所示。

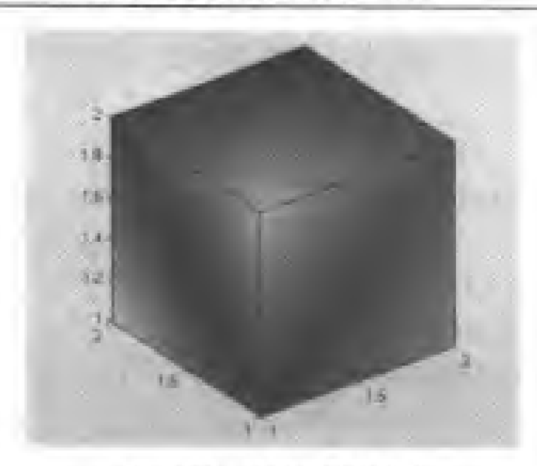


图 12-16　插值着色的效果

使用高级语法格式，用 x, y, z 和 c 参数可以进行同样的着色，其中，c 必须是一个 $m \times n \times 3$ 的数组，x, y 和 z 的维数为 $m \times n$。

### 12.2.3　控制面片着色的属性

表 12-3 概括了部分控制着色的面片函数（有光源时使用的函数除外）。

表 12-3　控制面片着色的属性

| 属　性 | 功　能 |
|---|---|
| CData | 用 x, y, z 数据指定每个小面或每个顶点的颜色 |
| CDataMapping | 指定是否将颜色数据进行比例化或直接用作颜色映射的索引 |
| FaceVertexCData | 用小面和顶点数据指定每个小面或顶点的颜色 |
| EdgeColor | 指定边的显示，包括隐藏、单色、顶点颜色确定的刻面颜色或顶点颜色确定的插值颜色 |
| FaceColor | 指定小面的显示，包括隐藏、单色、顶点颜色确定的刻面颜色或顶点颜色确定的插值颜色 |
| MarkerEdgeColor | 指定闭合图形的边缘色 |
| MarkerFaceColor | 指定闭合图形的填充色 |

### 12.2.4　面片边的着色

面片的每个小面都有包围它的边，边的着色与小面的着色方法基本相同，对应的也有单色、刻面和插值着色 3 种。

注意，只有在给每个顶点都指定颜色的情况下才能进行刻面着色和插值着色。对于刻面着色，MATLAB 把边起点的颜色作为边的颜色。所以，点的顺序很重要。

下面的例子用刻面着色方法对边和小面进行着色。

首先，创建一个方形面片。

```
v  = [0 0 0;1 0 0;1 1 0;0 1 0];
f = [1 2 3 4];
fvc = [1 0 0;0 1 0;1 0 1;1 1 0];
patch('Vertices',v,'Faces',f,'FaceVertexCData',fvc,...
      'FaceColor','flat','EdgeColor','flat',...
      'Marker','o','MarkerFaceColor','flat')
```

结果如图 12-17 所示。

Faces 属性的值[1 2 3 4]确定 MATLAB 连接顶点的顺序。这里，连接顺序是红、绿、洋红和黄。如果改变这个顺序，则结果会大不一样。例如，像下面这样指定 Faces 属性值。

```
f = [4 3 2 1];
```

将次序改变为黄、洋红、绿和红。注意，改变次序不仅仅改变边的颜色，还改变了面的颜色，面的颜色是指定的第 1 个顶点的颜色。改变次序后的图形效果如图 12-18 所示。

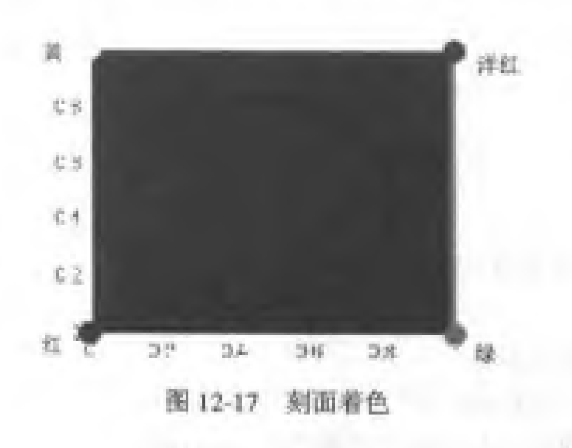



图 12-17　刻面着色

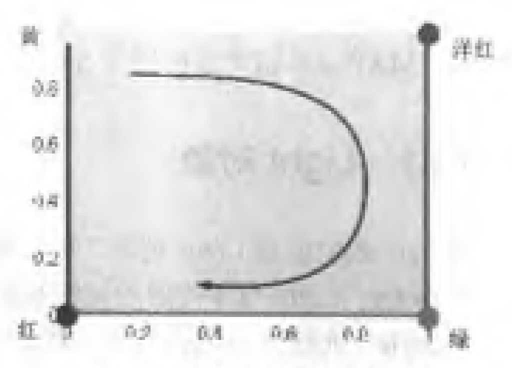



图 12-18　改变次序后的着色效果

# 第 13 章　光照与材质

添加光照使图形场景更加真实，这一点是通过模拟自然光下对象的明暗色调来实现的。这里，自然光也就是阳光。为了创建光照效果，MATLAB 定义了一个名为 Light 的图形对象。MATLAB 将光照应用于 Surface 和 Patch 对象。

## 13.1　Light 对象

用 light 函数创建 Light 对象。该对象有 3 个重要的属性，即

- Color　Light 对象投射的光线的颜色。
- Style　光线来自无穷远处（默认）还是附近。
- Position　对于无穷远处的光源，表示光的方向；对于附近的光源，表示光的位置。

Color 属性确定光的颜色。场景中对象的颜色由对象本身的颜色和光的颜色共同确定。

Style 属性确定光源是点光源（Style 属性值为 local）还是放在无穷远处的光源（Style 属性值设置为 infinite）。点光源从指定点上向各个方向发射，而放在无穷远处的光源从指定位置上发射镜面平行光。

Position 属性指定光源的位置，单位为坐标数据的单位。如果光源在无穷远处，则 Position 属性指定光源的方向。

光源会影响与 Light 对象同在一个坐标系中的 Surface 和 Patch 对象。这些对象有很多属性，利用这些属性，可以改变被光线照射时的外观显示。

## 13.2　光照命令

MATLAB 提供了几个用于设置光源位置和材质的命令，如表 13-1 中所示。

表 13-1　设置光源位置和材质的命令

| 命　令 | 功　能 |
|---|---|
| camlight | 相对于相机位置创建或移动 Light 对象 |
| lightangle | 在球坐标中创建或放置一个 Light 对象 |
| light | 创建一个 Light 对象 |
| lighting | 选择一种光照方法 |
| material | 设置对象的材质 |

## 13.3　给场景添加光照

下面的例子显示一个薄膜表面对象，用镜面光照射它，光的方向由向量[0 –2 1]定

义。连接该向量定义的空间点和坐标原点，得到一个方向。光就沿这个方向射向坐标原点。

```
membrane
light('Position',[0 -2 1])
```

创建一个 Light 对象可以激活很多与光照有关的属性，例如对象的散射光和反射光等。薄膜的光照效果如图 13-1 所示。

光照可以增强数学函数图形的表面效果。例如，下面使用 ezsurf 命令计算表达式 $\sin(\sqrt{x^2+y^2})+\sqrt{x^2+y^2}$ 在[−6π π]值域内的结果。

```
ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)',[ -6*pi,6*pi])
```

得刻面图如图 13-2 所示。

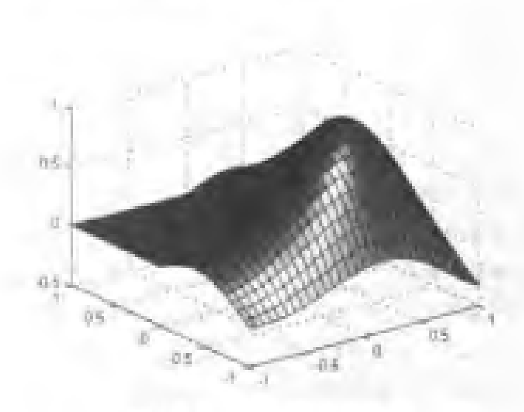


图 13-1　薄膜的光照效果

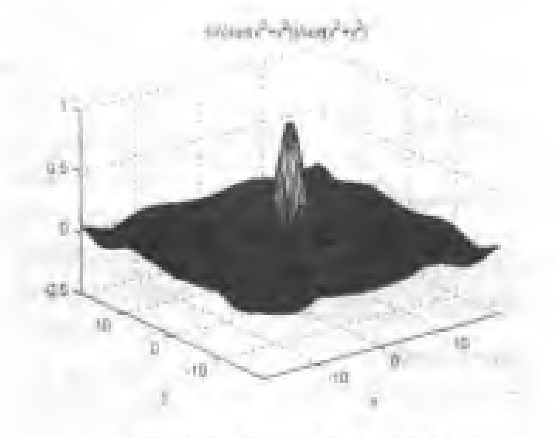


图 13-2　数学函数的刻面图

现在用 lightangle 命令添加光照。

```
view(0,75)
shading interp
lightangle(-45,30)
set(gcf,'Renderer','zbuffer')
set(findobj(gca,'type','surface'),...
   'FaceLighting','phong',...
      'AmbientStrength',.3,'DiffuseStrength',.8,...
      'SpecularStrength',.9,'SpecularExponent',25,...
      'BackFaceLighting','unlit')
```

效果如图 13-3 所示。

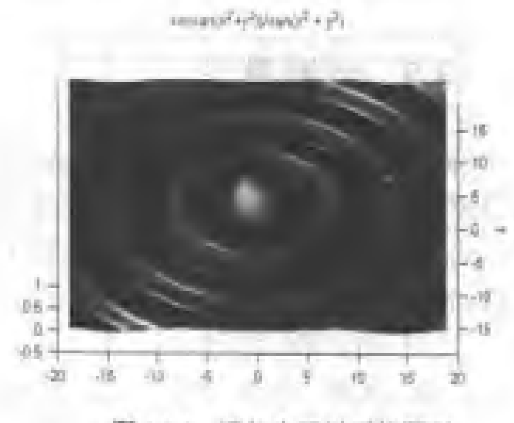


图 13-3　添加光照以后的图形

## 13.4　影响光照效果的属性

Light 对象本身是不可见的，但是可以在包含 Light 对象的坐标系中任何面片和表面对象上看到光照的效果。有很多函数可以生成这些对象，包括 surf,mesh,pcolor,fill,fill3,surface 和 patch 函数等。

通过设置不同的 Axes,Light,Patch 和 Surface 对象属性，可以控制光照效果。所有属性都

有默认值，可以生成需要的结果。但是，也可以通过调节这些属性值来获得指定的效果。这些属性及其控制效果如表 13-2 所示。

表 13-2 控制光照效果的属性

| 属　性 | 效　果 |
| --- | --- |
| AmbientLightColor | 是一个坐标系属性，它指定场景中背景光的颜色。背景光没有方向，对所有对象的影响都是相同的。背景光效果只在坐标系中有可见光对象时才会出现 |
| AmbientStrength | 是一个面片和表面属性，它确定从对象反射来的光中 ambient 组分的强度 |
| DiffuseStrength | 是一个面片和表面属性，它确定从对象反射来的光中 diffuse 组分的强度 |
| SpecularStrength | 是一个面片和表面属性，确定从对象反射来的光中 specular 组分的强度 |
| SpecularExponent | 是一个面片和表面属性，确定指定镜面光大小的面片和表面属性 |
| SpecularColorReflectance | 是一个面片和表面属性，确定反射光被对象颜色或光源颜色着色的程度 |
| FaceLighting | 是一个面片和表面属性，确定计算对象表面上光照效应的方法，包括没有光照、刻面、Gouraud 或 Phong 光照算法 |
| EdgeLighting | 是一个面片和表面属性，确定用于计算对象边上光照效应的方法，包括没有光照、刻面、Gouraud 或 Phong 光照算法 |
| BackFaceLighting | 是一个面片和表面属性，确定顶点法向指向远离相机的一侧时表面的颜色。这个属性对于区分对象的内表面和外表面很有用 |
| FaceColor | 是一个面片和表面属性，指定对象小面的颜色 |
| EdgeColor | 是一个面片和表面属性，指定对象边的颜色 |
| VertexNormals | 是一个面片和表面属性，包含对象各顶点的法向量。MATLAB 用法向量进行光照计算。MATLAB 会自动计算这些数据，也可以自己指定顶点法向 |
| NormalMode | 是一个面片和表面属性，确定如果改变对象数据或使用 VertexNormals 属性的当前值，是否重新计算顶点法向，如果指定 VertexNormals 的值，MATLAB 将该属性设置为 manual |

## 13.5 光照算法

给坐标系添加光照时，MATLAB 确定坐标系中面片和表面对象上的光照效果。有几种不同的方法可以用于计算对象边和面的着色，方法的选择与想获得的光照效果有关。

对于光照计算，MATLAB 提供了 3 种不同的算法。通过设置场景中每个面片和表面对象的 FaceLighting 和 EdgeLighting 属性可以进行选择。每个算法都会生成多少有些不同的结果：

- 刻面光照　对象的每个小面上生成均匀的颜色。选择这种方法查看平面对象。
- Gouraud 光照　计算顶点处的颜色，然后插值计算整个小面上的颜色。选择此法查看曲面对象。
- Phong 光照　在每个小面上进行顶点法向的插值，并计算每个像素的反射。选择此法查看曲面对象。Phong 光照生成的效果通常比 Gouraud 光照的好，但是更费时。

图 13-4 显示了应用几种光照方法时红球的显示效果。第 1 至第 4 个红球分别对应于没有光照、使用刻面光照、Gouraud 光照和 Phong 光照时的显示效果。

利用 lighting 命令，可以方便地设置光照方法。

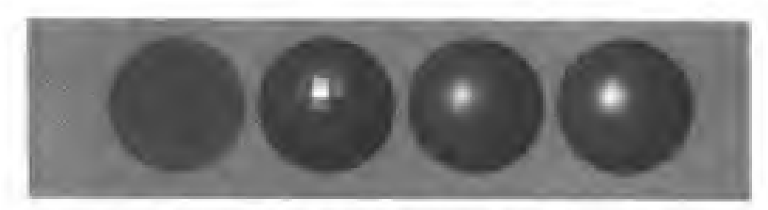


图 13-4　应用几种不同光照方法时红球的显示效果

## 13.6　图形对象的反射特性——材质

可以修改面片和表面对象的反射特性，从而改变在场景中应用光照时对象的显示外观。这些特性包括：

- 镜面反射和漫反射
- 环境光
- 镜面反射指数
- 镜面反射光的颜色
- 背面光照

可以组合使用这几种特性来生成特殊的显示结果。

### 13.6.1　镜面反射和漫反射

可以通过设置 SpecularStrength 和 DiffuseStrength 属性来控制对象表面镜面反射和漫反射的强度。图 13-5 演示了不同的设置效果。

### 13.6.2　环境光

环境光不是镜面光，它均匀地洒在场景中的所有对象上。只有在坐标系中有 Light 对象时环境光才可见。有两个属性可以控制环境光：AmbientLightColor 属性是一个用于坐标系的属性，设置颜色；AmbientStrength 属性是一个用于面片和表面对象的属性，确定特定对象上环境光的强度。

图 13-6 显示了 3 种不同颜色的环境光在不同强度下红球的显示效果。场景中有一个白光对象存在。

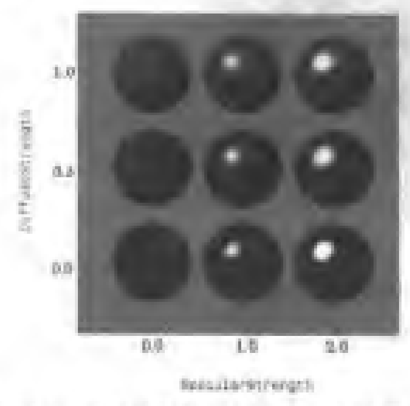



图 13-5　不同反射类型和强度下的设置效果

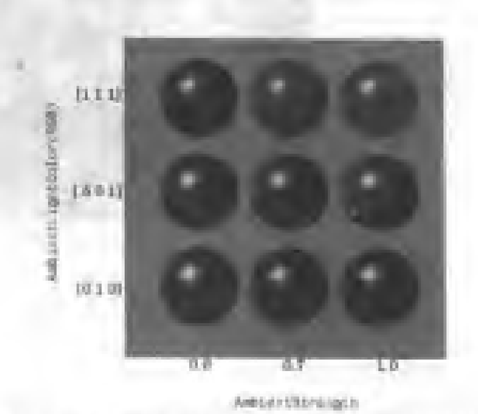



图 13-6　3 种不同颜色的环境光在不同强度下红球的显示效果

图 13-6 中，绿色([0 1 0])环境光对场景没有影响，因为绿光中没有红色组分。但是，RGB 值[.5 0 1]定义的颜色有红色组分，所以它会影响红球的显示效果，但它的影响效果没有白光[1 1 1]强烈。

### 13.6.3　镜面反射指数

镜面高光的大小与面片和表面对象的 SpecularExponent 属性有关，该属性的值界于 1 和 500 之间，一般对象的值界于 5 和 20 之间。图 13-7 演示了 SpecularExponent 属性取 3 个不同的值时白光照射下红球的显示效果。

### 13.6.4　镜面反射光的颜色

镜面反射光的颜色可以有一个变化范围，即从对象颜色与光源颜色的组合色变到只有光源颜色。面片和表面对象的 SpecularColorReflectance 属性控制这个颜色。图 13-8 显示了 SpecularColorReflectance 属性的值从 0 变到 1 时白光照射下红球的显示效果，0 对应于对象和光源的组合色，1 对应于光源的颜色。

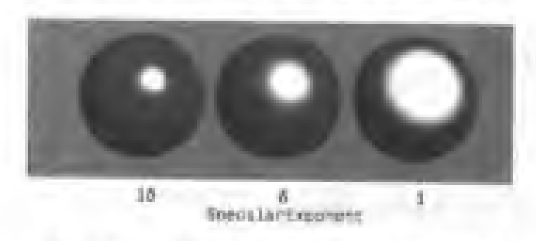



图 13-7　镜面反射指数对光照效果的影响

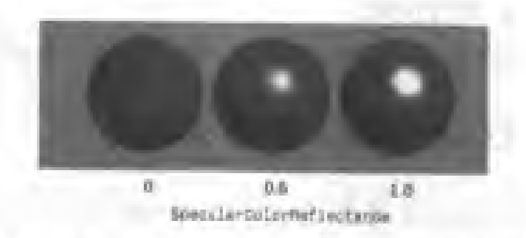



图 13-8　镜面反射光的颜色对显示效果的影响

### 13.6.5　背面光照

背面光照可用于显示对象内表面和外表面的差别。图 13-9 中被切开的柱面演示了这种背面光照效果。

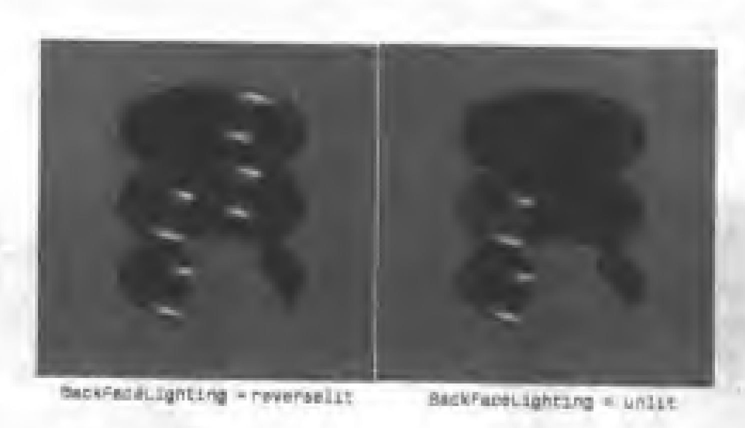



图 13-9　背面光照对显示效果的影响

BackFaceLighting 属性的默认值是 reverselit。这个设置将顶点法向指向相机，使内表面将光线反射到相机。将 BackFaceLighting 属性设置为 unlit，通过将顶点法向指向背离相机的

方向来取消光照。

还可以用 BackFaceLighting 属性删除闭合对象的边缘效应。当 BackFaceLighting 属性设置为 reverselit 时会发生这种效应，闭合对象边缘的像素会加亮，就好象它们的顶点法向面对相机一样。图 13-10 显示了这种边缘效应，将 BackFaceLighting 属性设置为 lit 后，边缘效应消失，如图 13-11 所示。

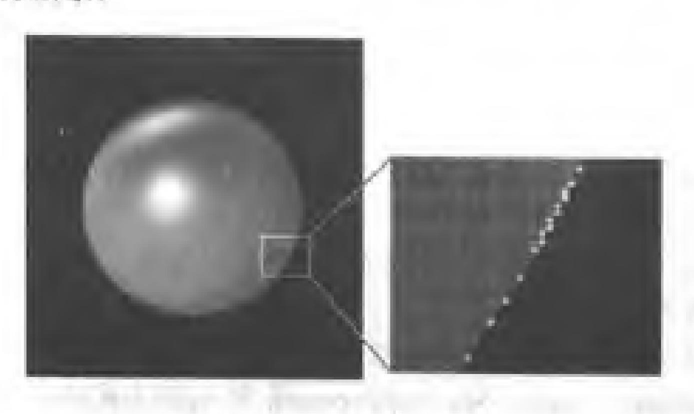



图 13-10　有边缘效应

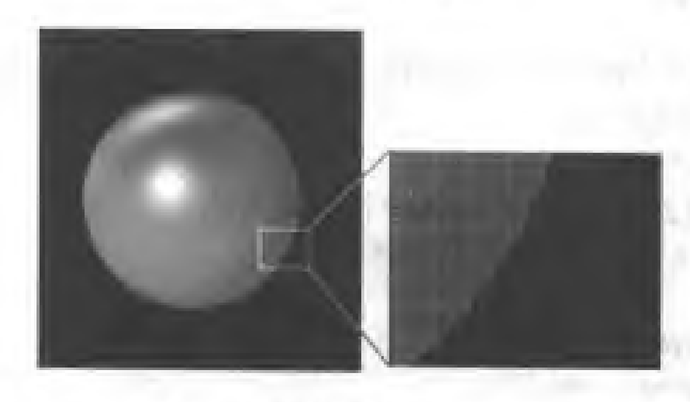



图 13-11　无边缘效应

### 13.6.6　material 函数

使用 material 函数也可以设置表面或面片的材质。该函数的语法格式为

```
material shiny
material dull
material metal
material([ka kd ks])
material([ka kd ks n])
material([ka kd ks n sc])
material default
```

对于各语法格式，有下面的描述：

- material shiny　镜面反射光的强度比漫反射光和环境光的要高得多，镜面光的颜色只决定于光源的颜色。
- material dull　主要进行漫反射，没有镜面反射，但是反射光的颜色只与光源有关。
- material metal　镜面反射很强，环境光和漫反射光较弱，反射光的颜色与光源和对象的颜色都有关系。
- material([ka kd ks])　设置对象的环境光、漫反射光和镜面光的强度。
- material([ka kd ks n])　设置环境光、漫反射光和镜面光的强度，以及对象的镜面反射指数。
- material([ka kd ks n sc])　设置环境光、漫反射光和镜面光的强度，以及对象的镜面反射指数和镜面反射光的颜色。
- material default　将环境光、漫反射光和镜面光的强度，以及对象的镜面反射指数和镜面反射光的颜色设置为默认值。

注意，material 命令设置坐标系中所有表面和面片对象的 AmbientStrength、DiffuseStrength、SpecularStrength、SpecularExponent 和 SpecularColorReflectance 属性。坐标系中必须有一个可见的 Light 对象。

### 13.6.7　一个例子

下面的例子创建一个球体和一个立方体，演示不同光照属性对应的显示效果。变量 vert 和 fac 用 patch 函数定义立方体。

在命令窗口输入下面的代码：

```
vert=[1 1 1;1 2 1;2 2 1;2 1 1;1 1 2;1 2 2;2 2 2;2 1 2];
fac=[1 2 3 4;2 6 7 3;4 3 7 8;1 5 8 4;1 2 6 5;5 6 7 8];
sphere(36);
h = findobj('Type','surface');
set(h,'FaceLighting','phong',...
     'FaceColor','interp',...
     EdgeColor',[.4 .4 .4],...
     BackFaceLighting','lit')
hold on
patch('faces',fac,'vertices',vert,'FaceColor','y');
light('Position',[1 3 2]);
light('Position',[ -3 -1 3]);
material shiny
axis vis3d off
hold off
```

立方体所有小面的 FaceColor 属性设置为 yellow。sphere 函数创建一个球形表面，用 findobj 函数查找 Type 属性为 surface 的对象，从而可以获取该表面的句柄。light 函数定义两个无穷远处的白光对象，它们的方向由 Position 向量指定。这些向量用坐标[x,y,z]定义。默认时，面片对象使用 flat Facelighting 属性值增强每个边的可见性。表面对象使用了 phong

FaceLighting 属性值，因为生成了最平滑的光照插值效果。material shiny 命令会影响立方体和球体的反射属性。

因为球体是闭合的，所以 BackFaceLighting 属性从默认设置变成了正常光照，删除了不必要的边缘效应。

生成的图形效果如图 13-12 所示。

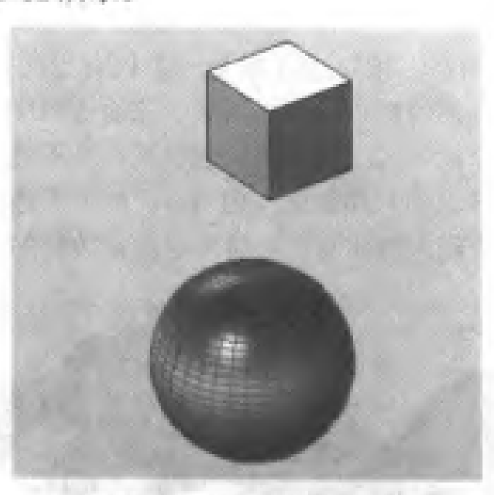


图 13-12　光照属性设置效果

# 第 14 章　透　明　性

将图形对象设置为半透明在三维可视化中是一个很有用的技巧。它使用户不仅能看到对象的外观，还能看到对象内部的结构特征。图形对象的透明度决定视线穿过对象的程度。可以指定透明度的一个连续变化范围，即从完全透明到完全不透明。

支持透明性的对象有图像、面片和表面。图 14-1 演示了透明效果。图中，绿色的等值面是半透明的，可以透过它看到内部的流锥。流锥是向量数据的一种表现方式。

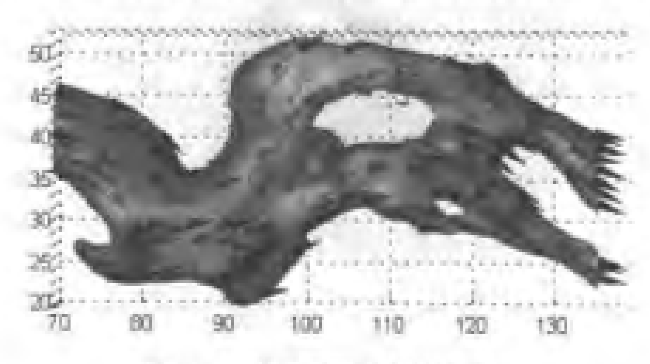



图 14-1　设置图形对象的透明性

注意，使用透明性，系统中必须有 OpenGL 库。如果该库存在，MATLAB 会自动使用它们。

## 14.1　使对象透明

### 14.1.1　alpha 值

透明值界于 0 和 1 之间，常用 alpha 值进行引用。当 alpha 值等于 0 时，表示完全透明；alpha 值等于 1 时，表示完全不透明。

MATLAB 中处理透明性的方式与处理颜色的方式有些类似：

面片和表面可以定义小面和边的 alpha 值，或根据 alpha 查找表中的值使用刻面或插值透明度。

图像、面片和表面对象可以将 alpha 数据作为索引值查找 alpha 查找表中的透明度或直接用作 alpha 值。

坐标系对象定义 alpha 的范围，它控制对象数据向 alpha 值的映射。

图形窗口对象包含 alpha 查找表，它是一个 $m\times1$ 的数组，元素为 alpha 值。

### 14.1.2　与透明性相关的属性

表 14-1 中概括了一些控制对象透明性的属性。

表 14-1 控制对象透明性的属性

| 属 性 | 功 能 |
|---|---|
| AlphaData | 图形和表面对象的透明度数据 |
| AlphaDataMapping | 透明度数据映射方法 |
| FaceAlpha | 小面的透明度（只对面片和表面可用） |
| EdgeAlpha | 边的透明度（只对面片和表面可用） |
| FaceVertexAlphaData | 只对面片可用的 alpha 数据属性 |
| ALim | alpha 坐标限制 |
| ALimMode | alpha 坐标限制模式 |
| Alphamap | 图形窗口的 alpha 查找表 |

表 14-2 列出了可以简化 alpha 属性设置过程的 3 个函数。

表 14-2 可以简化 alpha 属性设置过程的函数

| 函 数 | 功 能 |
|---|---|
| alpha | 设置或查询当前坐标系中对象的透明度属性 |
| alphamap | 指定图形窗口的 alpha 查找表 |
| alim | 设置或查询坐标系的 alpha 限制 |

## 14.2 指定一个单独的透明度值

希望揭示不透明对象的内部结构时，指定一个单独的透明度值是很有用的。对于面片和表面，用 FaceAlpha 和 EdgeAlpha 属性指定小面和边的透明度，可用下面的例子演示。

如果试图生成无界水域内水下喷嘴喷射时的速度分布轮廓数据，用图形表示该数据的方法之一是创建一个等值面，演示什么地方的流动速率等于指定值。

```
[x y z v] = flow;
p = patch(isosurface(x,y,z,v, -3));
isonormals(x,y,z,v,p);
set(p,'facecolor','red','edgecolor','none');
daspect([1 1 1]);
view(3); axis tight; grid on;
camlight; lighting gouraud;
```

生成图 14-2。

下面的语句将等值面的 FaceAlpha 值设置为.5，给该等值面添加透明度，揭示等值面内部的结构。

```
alpha(.5)
```

结果如图 14-3 所示。

对于图像，下面的语句将 AlphaData 属性的值设置为.5。当 AlphaDataMapping 属性设置为 none 时，设置图像的 AlphaData 值会使整幅图像用指定的 alpha 值进行渲染。

```
alpha(.5)
```

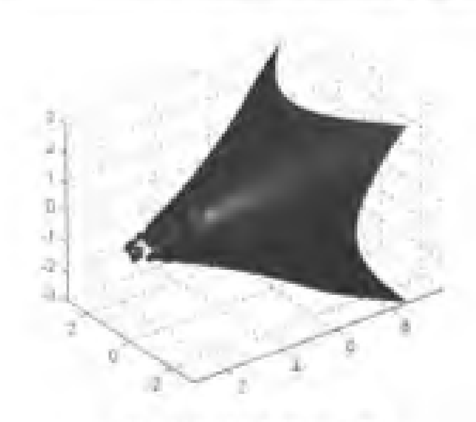


图 14-2　速度等值面图

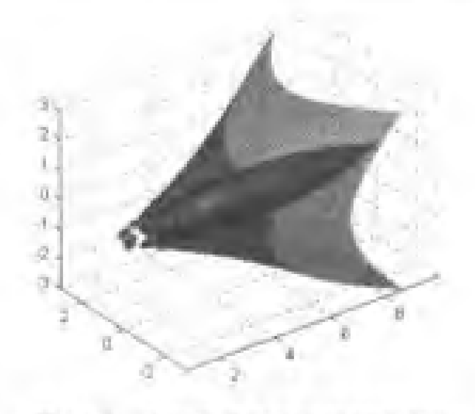


图 14-3　添加透明度以后的等值面图

## 14.3　将数据映射给透明度

alpha 数据与颜色数据类似，创建表面对象时，MATLAB 会将颜色数据数组中的每个元素映射到颜色查找表中的一种颜色。近似地，alpha 数据中的每个元素会映射到 alpha 查找表中的一个透明度值。

用 AlphaData 属性指定表面和图像的 alpha 值。对于面片对象，则使用 FaceVertexAlphaData 属性。

可以控制 MATLAB 如何用下面的属性解释 alpha 数据：

FaceAlpha 和 EdgeAlpha 属性　使得可以选择刻面或插值渲染方式。如果设置为单一透明度值，则 MATLAB 会将这个值应用于所有小面和边，而且不使用 alpha 数据。

AlphaDataMapping 和 ALim 属性　确定 MATLAB 如何将 alpha 数据映射到 alpha 查找表。默认时，MATLAB 会将 alpha 数据比例化到[0 1]范围内。

Alphamap 属性　包含实际的透明度值。

注意，描述颜色和透明度的一些属性的默认值还是有一些差别，因为，与颜色相比，透明度不是默认显示的。表 14-3 列出了这些差别。

表 14-3　颜色属性和 alpha 属性默认值的差异

| 颜色属性 | 默认值 | alpha 属性 | 默认值 |
| --- | --- | --- | --- |
| FaceColor | flat | FaceAlpha | 1 |
| CData | 等于 ZData | AlphaData 和 FaceVertexAlphaData | 1 |

### 14.3.1　alpha 数据数组的大小

为了使用非标量 alpha 数据，需要将 alpha 数据指定为一个数组，这个数组的大小可以确定为：

图形和表面的 CData。

面片的 FaceVertexAlphaData 属性中定义的小面数目或顶点数目。

一旦指定了大小合适的 alpha 数据数组，就可以选择要使用的小面和边的渲染方法。

### 14.3.2 将 alpha 数据映射到 alpha 查找表

可以用 AlphaDataMapping 属性控制 alpha 数据向 alpha 查找表的映射。有 3 种可能的映射：

none 将 alpha 数据中的值解释为透明度值（数据值必须界于 0 和 1 之间，或者是 0 或 1）。这是默认的映射方法。

scaled 比例化映射。与颜色的比例化映射原理相同，可以参照。

direct 直接将 alpha 数据作为索引值映射到 alpha 查找表。

### 14.3.3 示例——将数据映射到颜色或透明度

下面显示一个二元函数的刻面图，颜色通过建立与 z 数据梯度的映射关系得到。在命令窗口键入下面的命令行：

```
[x,y] = meshgrid([-2:.2:2]);
z = x.*exp(-x.^2-y.^2);
surf(x,y,z,gradient(z)); axis tight
```

生成图 14-4。

可以按照相同的映射方法得到透明度。即

```
surf(x,y,z,'FaceAlpha','flat',...
    'AlphaDataMapping','scaled',...
    'AlphaData',gradient(z),...
    'FaceColor','blue');
axis tight
```

添加透明度以后的表面图如图 14-5 所示。

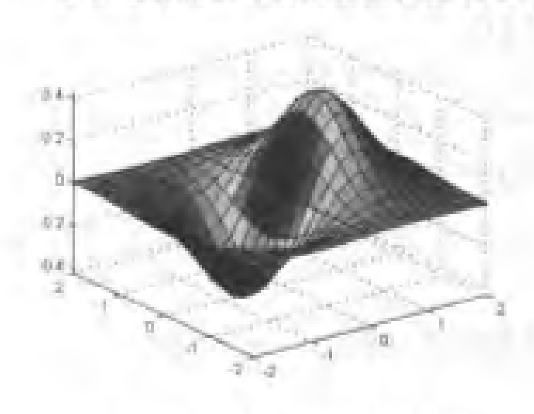


图 14-4 一个二元函数的刻面图

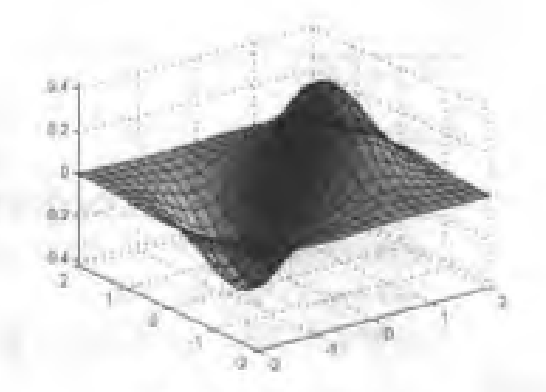


图 14-5 添加透明度以后的刻面图

## 14.4 选择一个 alpha 查找表

alpha 查找表只是一个数组，数组的值界于 0 和 1 之间，数组的大小可以是 $m\times1$ 或 $1\times m$。默认的 alpha 查找表包含 64 个值，这些值在 0 和 1 之间线性变化，如图 14-6 所示。用下面

的语句绘图。

```
plot(get(gcf,'Alphamap'))
```

alpha 查找表将最小的 alpha 数据值显示为完全透明，将最大的 alpha 数据值显示为不透明。alphamap 函数创建一些预定义的 alpha 查找表，还可以修改已经存在的映射关系。例如，

```
plot(alphamap('vup'))
```

生成图 14-7 所示的 alpha 查找表。

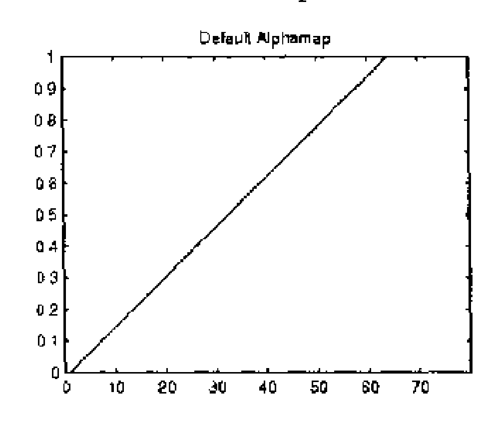



图 14-6　默认的 alpha 查找表

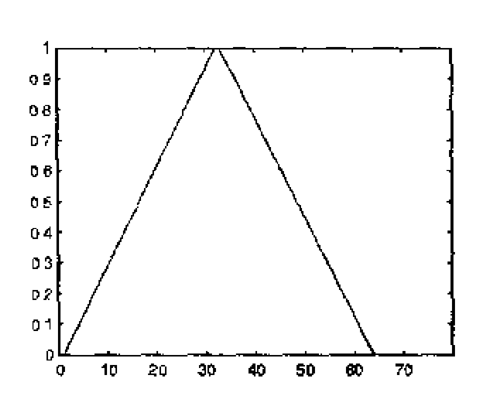


图 14-7　vup 查找表

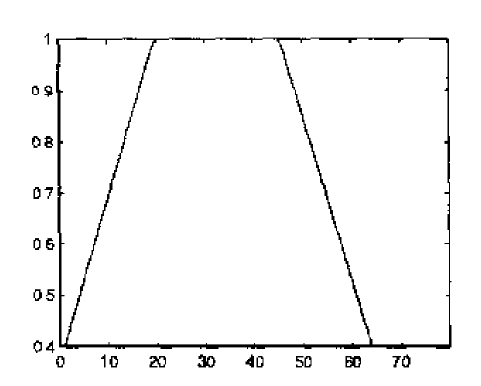


图 14-8　添加值以后的查找表

可以用 increase 或 decrease 选项转换这些值。例如，

```
alphamap('increase',.4)
```

把值.4 添加到当前图形窗口 alpha 查找表中的所有值上。重新绘出 alpha 查找表可以看出这种改变。

```
plot(get(gcf,'Alphamap'))
```

生成图 14-8。

下面的例子用切片面板检查体数据。绘切片面板时，直接将颜色数据作为 alpha 数据，并采用了 alpha 查找表 rampdown。按照以下步骤进行演示。

（1）通过计算一个 3 元函数来创建体数据。

```
[x,y,z] = meshgrid(-1.25:.1: -.25, -2:.2:2, -2:.1:2);
v = x.*exp(-x.^2-y.^2-z.^2);
```

（2）创建切片面板，设置 alpha 数据等于颜色数据，并指定 FaceAlpha 属性值为 interp。

```
h = slice(x,y,z,v,[ -1 -.75 -.5],[],[0]);
alpha('color')
set(h,'EdgeColor','none','FaceColor','interp',...
    'FaceAlpha','interp')
```

（3）载入 alpha 查找表 rampdown，将表中的每个值增加.1，以达到所需要的透明度。指定 hsv 颜色查找表。

```
alphamap('rampdown')
```

```
alphamap('increase',.1)
colormap(hsv)
```

这个 alpha 查找表使得函数的最小值（0 附近）用最小的透明度表示，而最大值用最大的透明度表示。这使得观察图形时既可以透过切片面板，同时又可以保留 0 周围的数据。图形效果如图 14-9 所示。

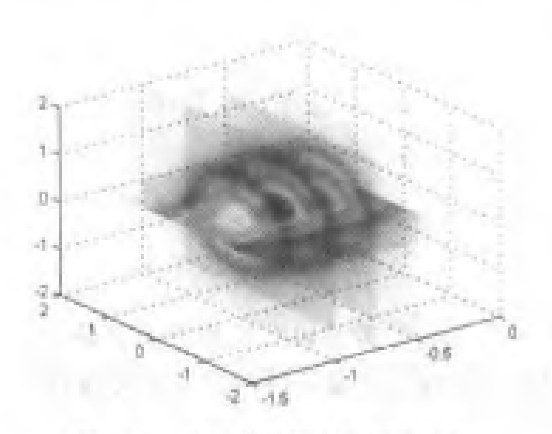


图 14-9 在切片面板中使用透明度

# 第 15 章 交互操作

所谓交互，指的是图形绘制到图形窗口中以后，可以用鼠标选中它，然后进行编辑，如进行平移、旋转、缩放、滚动和漫游等操作。交互功能是图形系统的关键功能，没有交互功能的图形系统是无趣的，也是没有生命力的。

## 15.1 视点和相机

本节介绍如何定义不同的视图参数，以获取想要的视图效果。通常，视图技术应用于三维图形或模型。MATLAB 视图设置主要包括两个方面的内容：

- 观察场景的视点位置；
- 设置坐标轴方向上的显示比率来控制要显示的对象的形状。

### 15.1.1 用方位角和仰角设置视点

用 MATLAB 可以控制图形在坐标系中的显示方向。可以将视点、视图目标、方向和景深这样一些视图特性通过一系列图形属性进行控制。可以直接给这些属性指定值，也可以用 view 命令或 MATLAB 自动选择的属性定义合理的视图。

view 命令用相对于坐标原点的方位角和仰角指定视点。方位角是 $x$-$y$ 平面上的极面角，正值表示逆时针方向旋转。仰角是相对于 $x$-$y$ 平面向上（正值）或向下（负值）的角度。

MATLAB 根据图形是二维还是三维自动选择视点：

- 对于二维图，默认时方位角为 0°，仰角为 90°；
- 对于三维图，默认时方位角为-37.5°，仰角为 30°。

现在举一个例子介绍怎样用方位角和仰角指定视点。下面的代码是创建一个三维表面图，并以默认的三维视图形式显示它。

```
[X,Y] = meshgrid([-2:.25:2]);
Z = X.*exp(-X.^2 -Y.^2);
surf(X,Y,Z)
```

结果如图 15-1 所示。

然后设置视点，方位为 y 轴正向，仰角为 0，即

```
view([30 30])
```

视图效果如图 15-2 所示。

也可以通过指定一个负的仰角，将视点移到坐标轴原点的下方，即

```
view([-37.5 -30])
```

视图效果如图 15-3 所示。

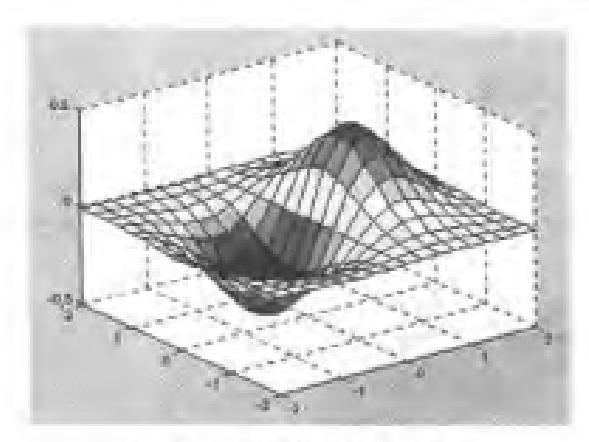


图 15-1 默认的三维视图显示

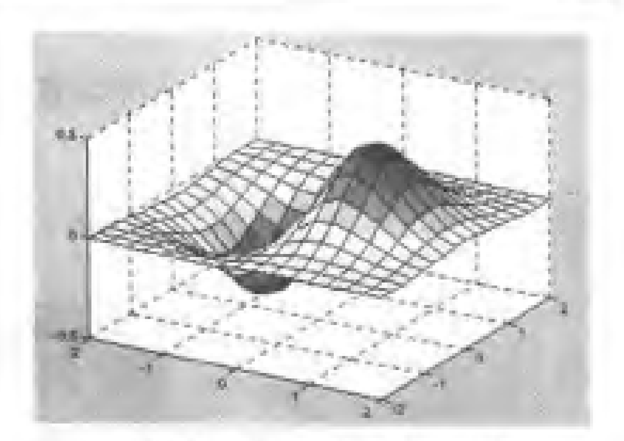


图 15-2 重设视点以后的视图效果

指定方位角和仰角在概念上很容易理解，但是在操作上有局限性。使用这种方法不能指定视点在 $z$ 轴正上方的情况，也不能对场景进行缩小和放大，或进行任意旋转和平移。

相比而言，MATLAB 的相机工具提供了更大的控制能力。下面介绍如何用相机属性控制视图。

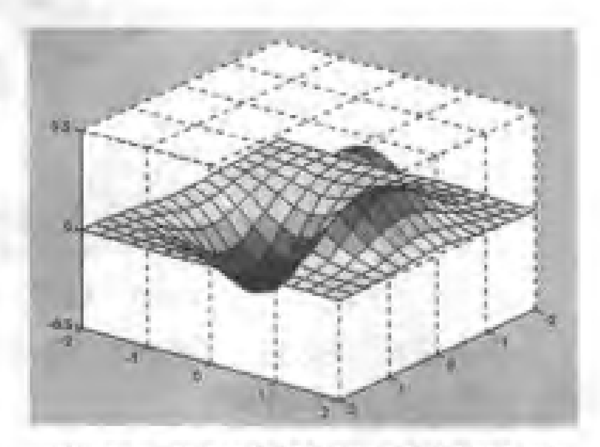


图 15-3 视点在坐标轴原点下方的视图效果

### 15.1.2 交互工具——相机

观察坐标系中显示的图形对象时，实际上是在空间中的某个特定位置上观察场景，这个位置相对于场景有一定的方向。MATLAB 提供了功能上类似于变焦相机的工具，使用它可以控制 MATLAB 创建的场景图。

图 15-4 演示了如何用坐标系属性定义相机。

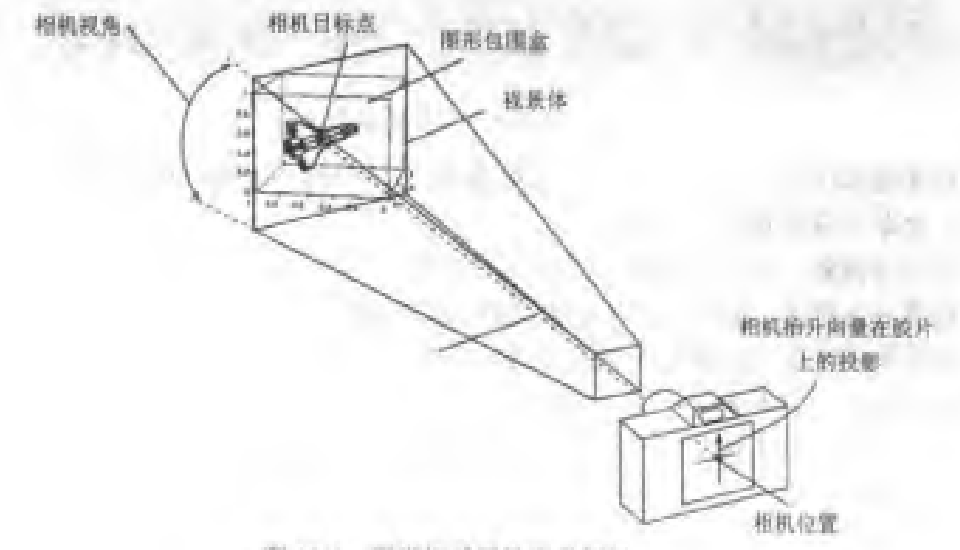



图 15-4 用坐标系属性定义相机

## 15.2　用相机工具条进行场景空间变换

使用相机工具条，可以进行很多视图交互操作。在图形窗口中显示三维图形以后，在图形窗口的“View”菜单中选择“Camera Toolbar”选项，可显示相机工具条，如图 15-5 中红框内所示。

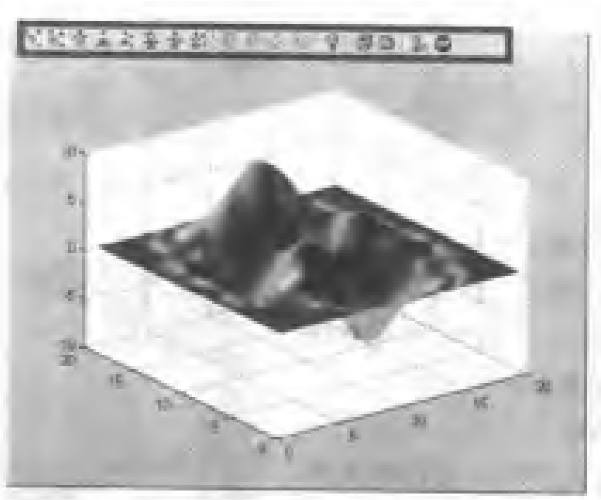


图 15-5　在图形窗口中显示相机工具条

工具条显示出来以后，选择相应的命令按钮，然后在图形场景上按住鼠标键以后进行拖拉，场景会进行相应的变换。

### 15.2.1　相机工具条

如图 15-6 所示，工具条包括下面几部分内容：




图 15-6　相机工具条

- 相机移动控制工具按钮　用于对场景进行平移、旋转和缩放等操作。也可以从“Tools”菜单中获取相应的选项。
- 主轴选择器　用于选择向上的坐标系主轴。
- 场景光照开关　控制场景中光照的打开和关闭。
- 投影类型按钮　选择正交投影或透视投影。
- 重设按钮　将场景重设为变换前的状态。
- 停止按钮　停止动画等。

### 15.2.2　变换主轴

场景的主坐标轴定义屏幕中方向向上的坐标轴。例如，MATLAB 表面图将 $z$ 轴正向指

定为向上的方向。

如果数据是相对于指定坐标轴定义的，则指定主坐标轴很有用。z 是默认的主坐标轴，因为它与默认的 MATLAB 三维视图相匹配。

可以用 3 种相机工具选择主轴和相关运动。屏幕上，旋转轴由一根垂直线和一根水平线确定，这两根线都穿过 CameraTarget 属性定义的点，并且与主轴平行和垂直。

例如，当主轴是 z 轴时，移动时会出现下面的情况：

- 一根垂线穿过相机目标对象，该垂线与 z 轴平行。
- 一根水平线穿过相机目标对象，它与 z 轴垂直。

这表示场景沿圆弧移动，圆弧中心在相机目标对象上。图 15-7 演示了 z 主轴的旋转轴。

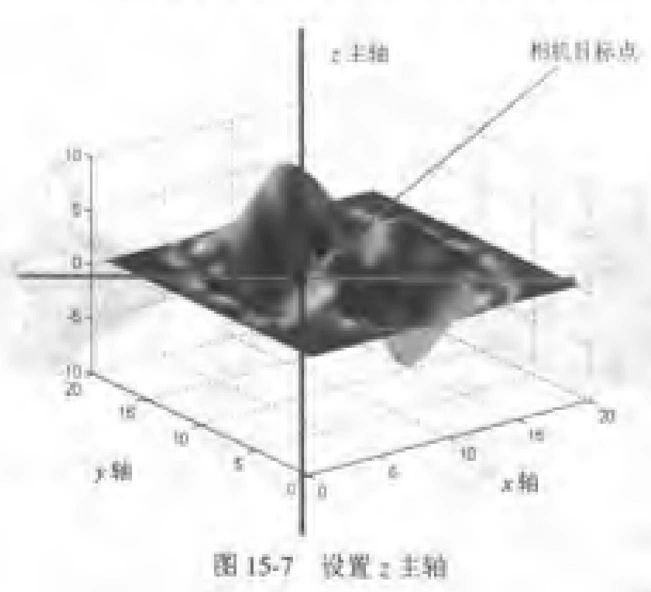



图 15-7 设置 z 主轴

### 15.2.3 盘旋

创建图形时，MATLAB 用一个各坐标轴方向上的显示比率显示图形，使它刚好装满图形窗口。对于三维图形操作，这可能不会生成最优的结果，因为在场景周围移动相机时可能会导致图形变形。为了避免可能的变形，最好转换为三维可视化模式（在命令行中使用 axis vis3d 命令）。使用相机工具条时，MATLAB 会自动转换到三维可视化模式，但是会首先给出图 15-8 所示的警告对话框。这个对话框只显示一次。

默认时，盘旋相机操作会使相机绕 z 轴旋转。可以用主轴选择器选择绕 x, y, z 或自由坐标轴旋转。没有使用主轴时，可以绕任意坐标轴旋转。

盘旋相机操作会在保持 CameraTarget 属性值固定的同时改变 CameraPosition 属性的值。相机盘旋操作的示意如图 15-9 所示。注意，相机始终对准目标点。

与盘旋有关的还有场景光的盘旋。场景光是一个光源，默认时，它放在相机右侧。盘旋场景光会改变光与相机位置之间的偏移量，并相应地改变场景的光照效果。只有一个场景光，可以用 light 命令添加其他的光。

在相机工具条中使用和按钮，可以实现相机盘旋和场景光盘旋。对图 15-5 中的图形进行这两种操作以后，得图 15-10 和图 15-11。

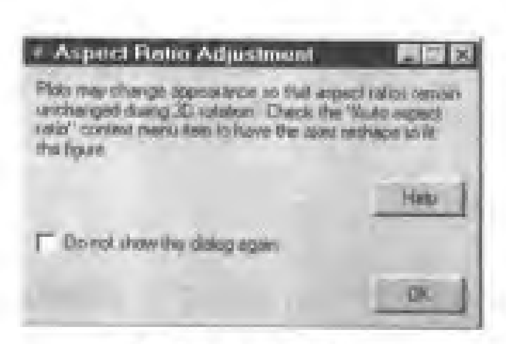



图 15-8　警告对话框

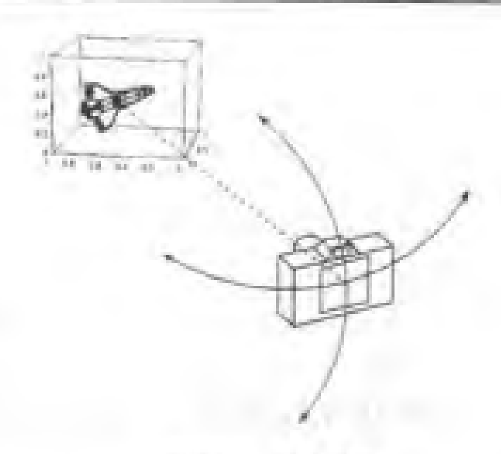


图 15-9　相机盘旋

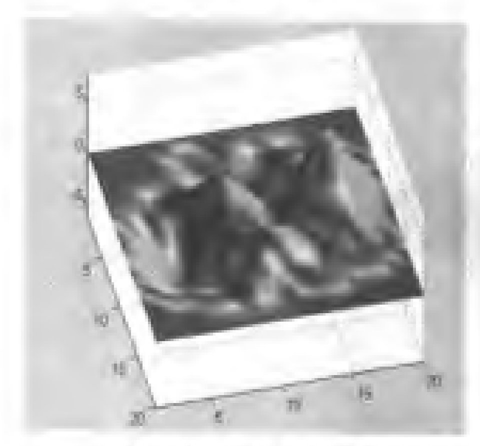


图 15-10　相机盘旋效果

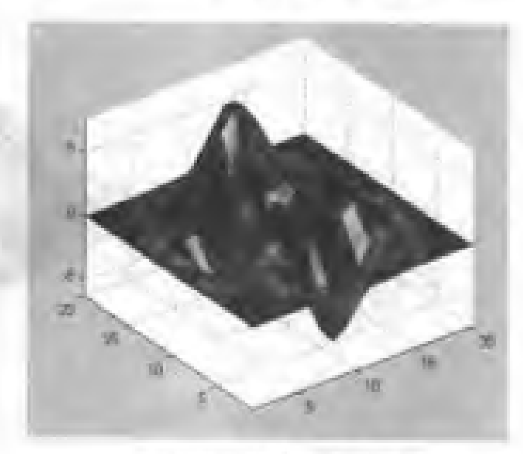


图 15-11　场景光盘旋效果

在工具条中单击按钮，可以关闭场景光。盘旋场景光是通过改变 light 对象的 Position 属性来移动场景光的。

### 15.2.4　平移

场景的平移变换可以用两种方法实现。一种是对场景本身进行平移，另一种是平移相机。前一种好理解。后一种就好比我们坐在汽车上看窗外的世界，窗外的花花草草并没有移动，但我们感觉它们在移动。MATLAB 对这两种平移方法都支持。

**1．平移目标点**

此时相机是固定的，而相机对准的场景中的目标点是移动的，如图 15-12 所示。默认时，这个点绕 $z$ 轴盘旋。可以用主轴选择器选择 $x, y, z$ 或自由轴旋转。

移动场景中的点是通过改变 CameraTarget 属性实现的。单击相机工具条中的按钮，然后在场景中单击并拖拉鼠标，可以平移场景。

**2．平移相机**

水平或垂直移动相机也能实现场景的移动，如图 15-13 中所示。单击相机工具条中的按钮，然后在场景中单击并拖拉鼠标，可以实现场景的平移。图 15-5 中场景平移的效果如图 15-14 所示。

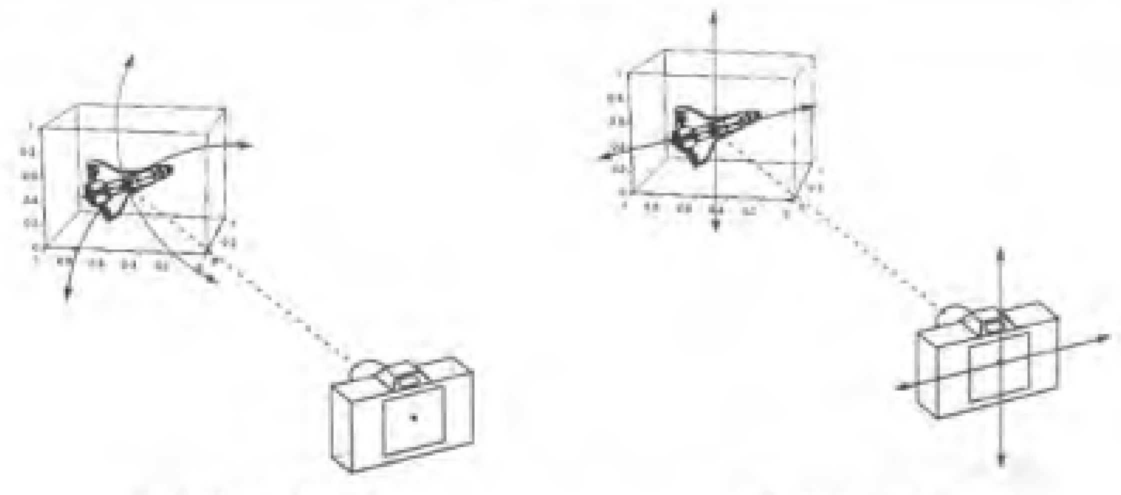


图15-12 平移目标点　　　图15-13 平移相机

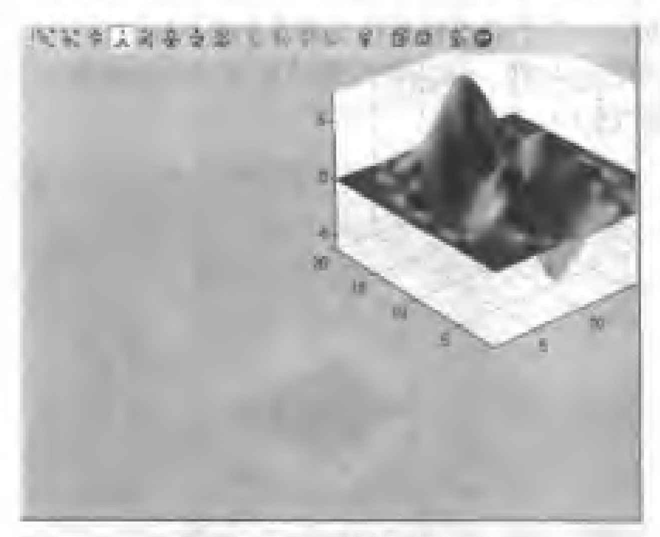


图15-14 场景平移

### 15.2.5 缩放

场景缩小和放大也可以通过两种方法来实现。一种是前后移动相机，另一种是改变相机的视角。

**1. 前后移动相机**

将光标向上或向右移动可以使相机向场景方向移动。将光标向下或向左移动可以使相机背离场景方向移动。使相机移动时穿过场景中的对象到达另一端也是可以的。

对应的图形属性会沿相机位置和相机目标位置的连线移动相机，如图15-15所示。

单击相机工具条中的按钮，然后在场景中单击并拖拉鼠标，可以缩放场景。

**2. 调整相机视角**

缩放操作可以通过改变CameraViewAngle属性，即通过调整相机视角来实现。属性值越大，场景看起来越小；反之场景看起来越大。调整相机视角缩放场景的原理如图15-16所示。

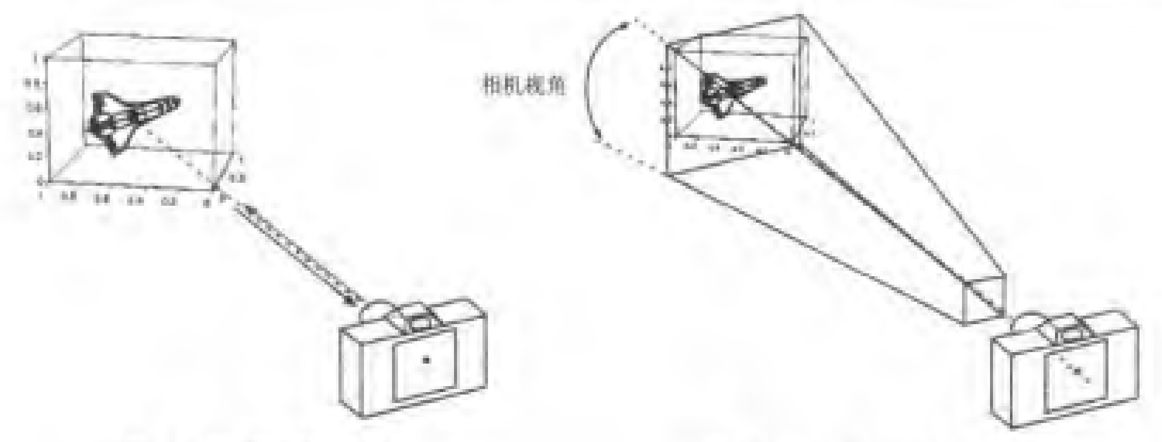



图 15-15　前后移动相机　　　　图 15-16　调整相机视角

在场景上按下鼠标以后，向上或向右拖拉会放大场景，向下或向左拖拉会缩小场景。缩放操作不会移动相机，所以不能使视点穿过场景中的对象。

单击相机工具条中的按钮，然后在场景中单击并拖拉鼠标，可以缩放场景。图 15-5 中场景缩小的效果如图 15-17 所示。

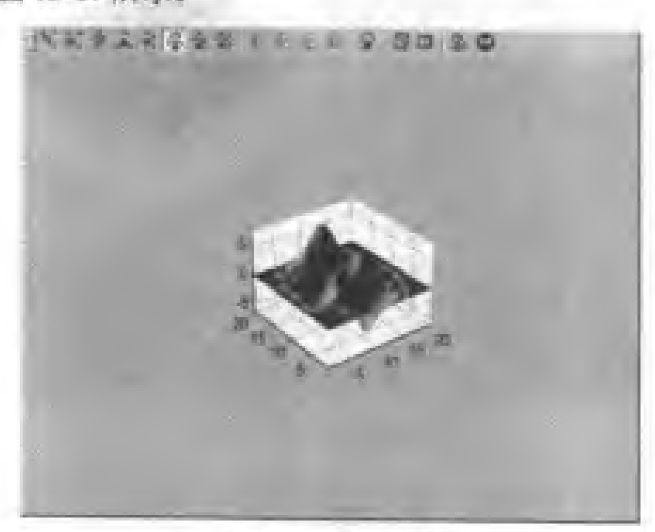


图 15-17　缩小场景

### 15.2.6　滚动

相机滚动操作使相机绕坐标轴旋转，从而旋转屏幕上的视图，如图 15-18 所示。该操作会改变 CameraUpVector 属性。

单击相机工具条中的按钮，然后在场景中单击并拖拉鼠标，可以滚动场景。图 15-5 中场景滚动后的效果如图 15-19 所示。

### 15.2.7　漫游

如图 15-20 所示，相机漫游会在相机目标点方向上移动相机，并且将相机目标点移动相同的量。该操作还会左右摇动相机。使用相机漫游，可以在整个场景中移动相机，穿过视轴上的所有对象。

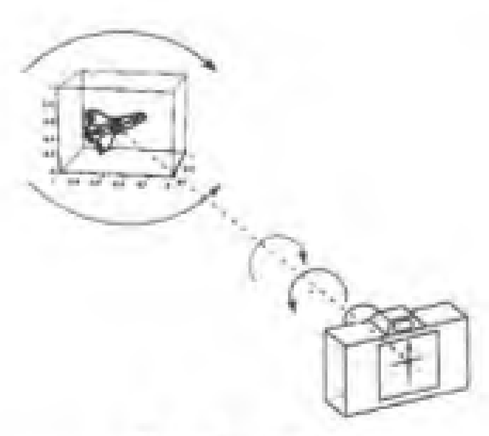


图 15-18 相机滚动

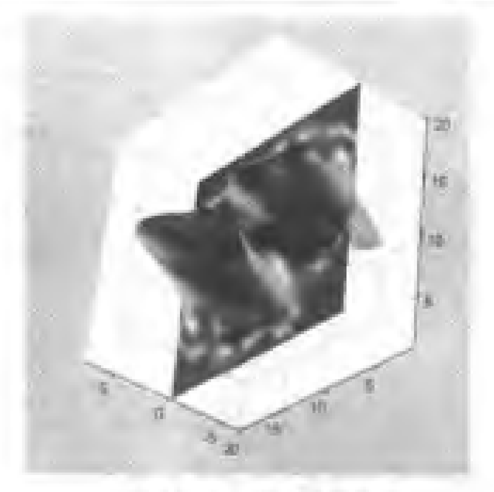


图 15-19 滚动视图效果

相机漫游在某种程度上与开车时目光始终注视前方相似。向右转时，场景中看到的对象出现在左侧。当视轴位于垂直主轴的平面上时，相机漫游的使用效果最好。例如，如果 z 轴是主轴，则应该将相机放在与相机目标点相同的高度上。此时趋近和远离的移动都会保持在相同的 z 值上。使用相机漫游以前将图形缩小效果会比较好一些。

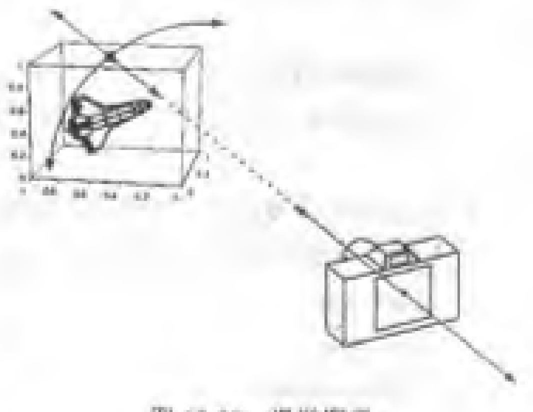


图 15-20 漫游图示

相机漫游操作对 CameraPosition 和 Camera Target 属性都会有所修改，以保持它们之间的距离不变。

单击相机工具条中的按钮，然后在场景中单击并拖拉鼠标，可以实现场景漫游。

## 15.3 用与相机有关的函数实现场景空间变换

MATLAB 提供了一些与相机有关的函数，利用这些函数也可以实现场景的空间变换，并且与使用相机工具条相比，使用函数更灵活，可以实现更多的功能。

### 15.3.1 与相机有关的函数

表 15-1 列出了可以进行相机操作的一系列函数。关于各函数的语法和用法，可以参见 MATLAB 帮助文档，这里不详细介绍。后面举了两个用函数进行场景变换的例子，中间用到了表中的部分函数。

表 15-1 与相机有关的函数

| 函 数 | 功 能 |
|---|---|
| camdolly | 移动相机位置和目标点 |
| camlookat | 查看指定的目标 |
| camorbit | 绕目标点盘旋相机 |

续表

| 函　数 | 功　能 |
|---|---|
| campan | 绕相机位置旋转相机目标点 |
| campos | 设置或获取相机位置 |
| camproj | 设置或获取投影类型 |
| camroll | 绕视轴旋转相机 |
| camtarget | 设置或获取相机目标点的位置 |
| camup | 设置或获取相机仰角的值 |
| camva | 设置或获取相机方位角的值 |
| camzoom | 用相机进行场景缩放操作 |

### 15.3.2　示例 1——平移图像

下面的例子演示如何用 camdolly 命令查看图像中的不同区域。该例用到了下面一些技巧：

- 用 ginput 函数获取图像中的位置坐标。
- 用 camdolly 数据坐标选项根据 ginput 函数获取的坐标将相机和目标点移动到新位置。
- 用 camva 函数放大图形并固定相机视角。

首先载入好望角地区的图像数据 cape 并通过设置相机视角放大图像。

```
load cape
image(X)
colormap(map)
axis image
camva(camva/2.5)
```

显示图 15-21。

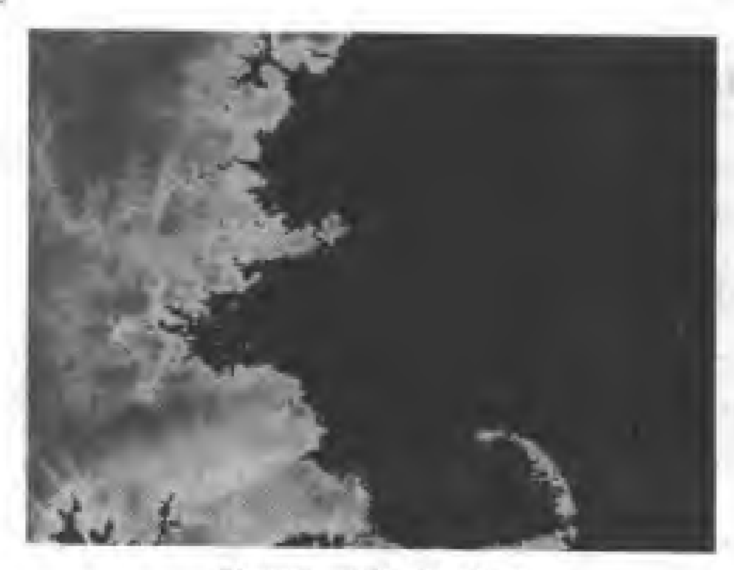


图 15-21　图像原来的位置

然后用 ginput 函数选择相机目标点和相机位置的 $x$ 和 $y$ 坐标值。

```
while 1
  [x,y] = ginput(1);
  if ~strcmp(get(gcf,'SelectionType'),'normal')
     break
  end
  ct = camtarget;
  dx = x - ct(1);
  dy = y - ct(2);
  camdolly(dx,dy,ct(3),'movetarget','data')
  drawnow
end
```

用 ginput 函数指定相机目标点和相机位置后的图像如图 15-22 所示。

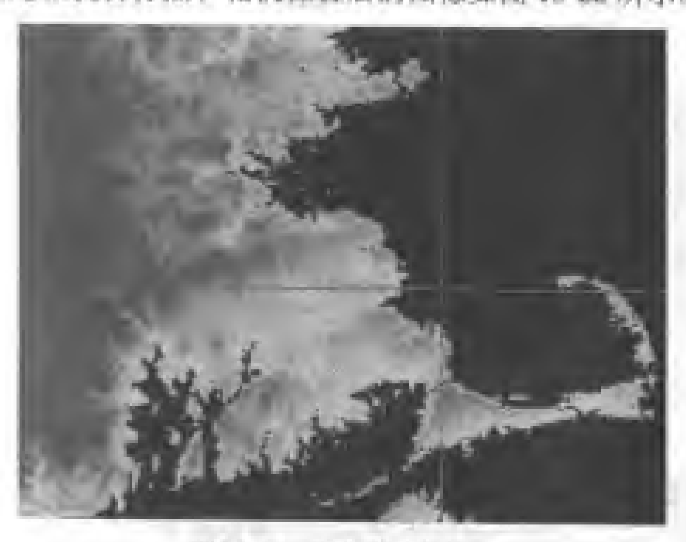


图 15-22 图像当前的位置

### 15.3.3 示例 2——穿越场景

穿越场景的效果通过在三维空间中移动相机来实现。下面的例子利用穿越效果观察等值面的内部结构。该等值面是用 wind 数据表示的速率向量场定义的，它表示北美洲上空的气流。

用到的技巧包括：

- 用等值面和流锥图演示三维区域内部气流的流动情况；
- 利用光照演示三维区域内部的等值面和锥体；
- 用流线定义相机穿越三维区域内部时的路径；
- 相机位置、目标点和光照等的坐标。

**1．体积数据图示**

第一步是画一个等值面并用流锥图表示气流的流动情况。

在绘流锥之前将沿各坐标轴方向的显示比率设置为[1,1,1]。在命令窗口添加下面的命令行：

```
load wind
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hpatch = patch(isosurface(x,y,z,wind_speed,35));
isonormals(x,y,z,wind_speed,hpatch)
set(hpatch,'FaceColor','red','EdgeColor','none');
[f vt] = reducepatch(isosurface(x,y,z,wind_speed,45),0.05);
daspect([1,1,1]);
hcone = coneplot(x,y,z,u,v,w,vt(:,1),vt(:,2),vt(:,3),2);
set(hcone,'FaceColor','blue','EdgeColor','none');
```

生成图 15-23。

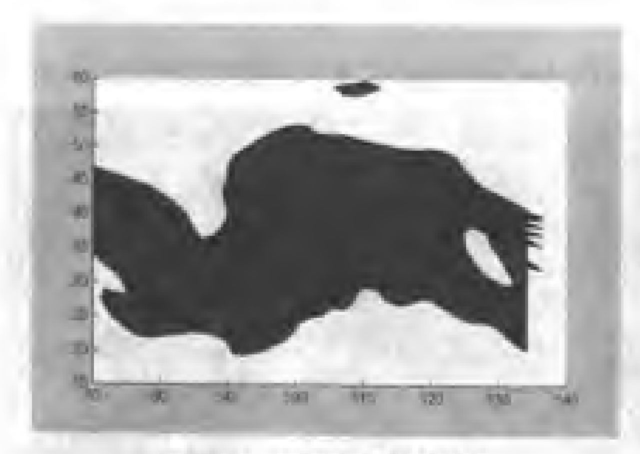


图 15-23　等值面叠加流锥图

2．设置视图

需要定义视图参数，以保证场景的正确显示。

- 选择透视投影，提供相机穿越等值面内部时的深度值。
- 设置相机视角为固定值，防止 MATLAB 自动调整角度以包围整个场景或者放大到需要的大小。

```
camproj perspective
camva(25)
```

3．指定光源

将光源放在相机处并修改等值面和流锥的反射特性，以增强场景的真实性。

- 在相机位置上创建一个光源，提供一个随相机在等值面内部移动的高光。
- 设置等值面的反射特性，通过使用高反射材质（将 SpecularStrength 和 DiffuseStrength 参数设置为 1）将等值面内部设置为比较深的颜色（将 AmbientStrength 参数设置为 0.1）。
- 将流锥的 SpecularStrength 属性设置为 1，使它们具备高反射特性。

在命令窗口添加下面的命令行：

```
hlight = camlight('headlight');
set(hpatch,'AmbientStrength',.1,...
        'SpecularStrength',1,...
        'DiffuseStrength',1);
```

```
set(hcone,'SpecularStrength',1);
set(gcf,'Color','k')
```

### 5．选择渲染器

因为本例使用光照，MATLAB 必须使用 zbuffer 渲染器或 OpenGL 渲染器。OpenGL 渲染器在显示快照时速度更快，但是此时需要设置 gouraud 光照，其效果没有设置 phong 光照时平滑。phong 光照可以与 zbuffer 渲染器一起使用。两种光照和渲染器的使用如下所示：

对于 OpenGL 渲染器：

```
lighting gouraud
set(gcf,'Renderer','OpenGL')
```

或者对于 zbuffer 渲染器：

```
lighting phong
set(gcf,'Renderer','zbuffer')
```

### 6．用流线描述相机移动路径

流线表示了向量场中的流动方向。本例使用单条流线的 $x$, $y$ 和 $z$ 坐标数据映射体中的移动路径。步骤包括：

- 创建一条起点为 $x = 80, y = 30, z = 11$ 的流线；
- 获取流线的 $x, y$ 和 $z$ 坐标数据；
- 删除流线。

在命令窗口添加下面的命令行：

```
hsline = streamline(x,y,z,u,v,w,80,30,11);
xd = get(hsline,'XData');
yd = get(hsline,'YData');
zd = get(hsline,'ZData');
delete(hsline)
```

### 7．完成穿越

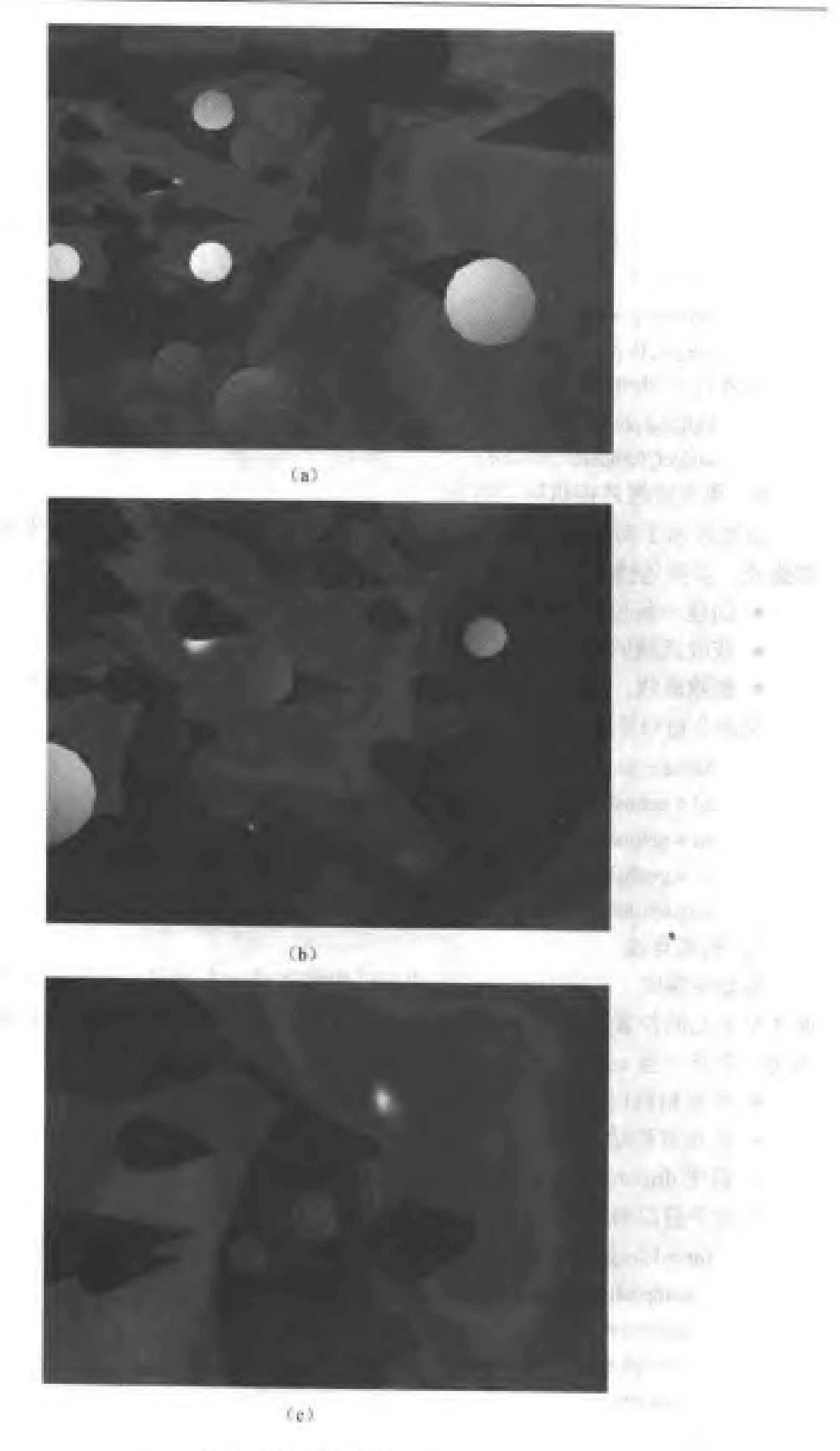
创建穿越时，沿同样的路径移动相机和目标点。本例中，目标点放在沿 $x$ 轴比相机位置前 5 个单元的位置上。另外，给目标点的 $x$ 坐标加了一个小值，以防止相机位置和目标点成为同一个点（当 $xd(n) = xd(n+5)$时会出现这种情况）：

- 更新相机位置和相机目标点，使它们都沿流线移动；
- 沿相机移动光照；
- 调用 drawnow 函数，显示每次移动的结果。

在命令窗口输入下面的命令行：

```
for i=1:length(xd)-50
  campos([xd(i),yd(i),zd(i)])
  camtarget([xd(i+5)+min(xd)/100,yd(i),zd(i)])
  camlight(hlight,'headlight')
  drawnow
end
```

生成图 15-24。图（a）、（b）和（c）分别为 $i$ 值等于 10,110 和 185 时的视图。



(a)

(b)

(c)

图 15-24　快照视图

### 15.3.4 低级相机属性

相机工具工作时是基于一组控制相机位置和方向的坐标轴属性的。通常，使用相机有关命令使得可以不必直接获取这些属性。这些属性如表15-2中所示。

表15-2 相机的低级属性

| 属性 | 功能 |
| --- | --- |
| CameraPosition | 指定视点的位置 |
| CameraPositionMode | 如果是automatic模式，MATLAB基于场景自动确定视点的位置；如果是manual模式，则自己指定视点位置 |
| CameraTarget | 指定坐标系中相机指向的位置。与CameraPosition属性一起使用，可以定义视图坐标 |
| CameraTargetMode | 如果是automatic模式，MATLAB将CameraTarget属性的值指定为坐标系中图形包围盒的中心；如果是manual模式，则自己指定位置 |
| CameraUpVector | 相机绕视图坐标的旋转用表示抬升方向的向量定义 |
| CameraUpVectorMode | 如果是automatic模式，对于二维视图，MATLAB沿y轴正向确定抬升向量中的方向，对于三维视图，则沿z轴正向确定。如果是manual模式，则直接指定方向 |
| CameraViewAngle | 指定"镜头"的视域。如果给CameraViewAngle属性指定一个值，MATLAB会使自动拉伸图形并使之充满图形窗口这一默认操作失效 |
| CameraViewAngleMode | 如果是automatic模式，MATLAB将视角调整到可以捕获整个场景的最小角度。如果是manual模式，则需要自己指定角度。将此属性设置为manual会使自动拉伸图形并使之充满图形窗口这一默认操作失效 |
| Projection | 选择正交投影或透视投影 |

## 15.4 投影

### 15.4.1 正交投影和透视投影

MATLAB支持用正交投影和透视投影两种投影方法显示三维图形。选择哪种方法与要显示的图形的类型有关。

- 正交投影　将视景体投影成一个成直角的平行六面体（即对边平行的盒子）。与相机的相对距离不影响对象的大小。当保持对象的实际大小和对象之间的角度很重要的时候，使用这种投影类型很有用。
- 透视投影　将视景体投影成一个台体，它就像顶部被截掉的金字塔。距离远时，对象会显得更小。希望显示真实对象的真实视图时，多使用这种投影方法。

默认时，MATLAB使用正交投影。可以用camproj命令设置投影类型。

图15-25中的两个图形显示了一辆自动卸货卡车和一个数学函数表面图，两个图形都使用了正交投影。

这两个图形使用透视投影时，显示效果如图15-26所示。

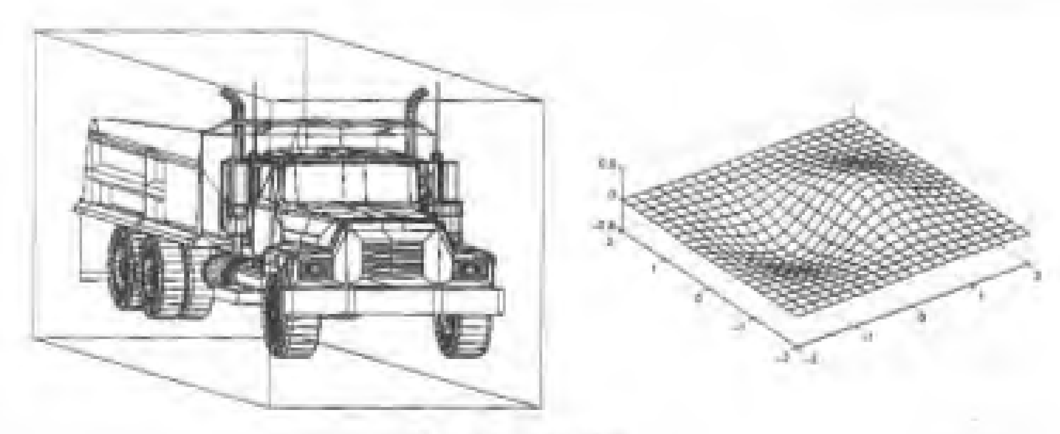


图 15-25　正交投影

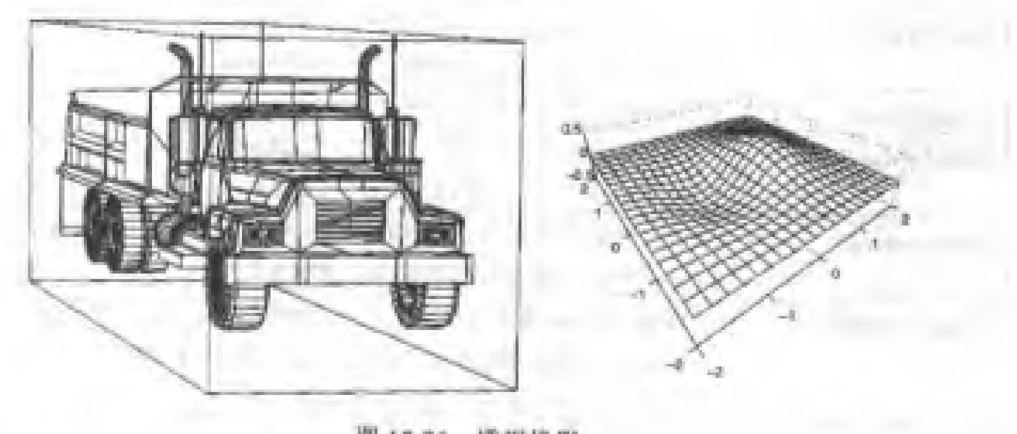


图 15-26　透视投影

### 15.4.2　投影类型和相机位置

默认时，MATLAB 会通过调整 CameraPosition,CameraTarget 和 CameraViewAngle 属性将相机放到场景中心，并包括坐标系中的所有图形对象。如果放置相机时图形对象在相机后面，则坐标系的 Projection 属性和图形窗口对象的 Renderer 属性都会影响场景的显示。表 15-3 概括了投影类型与渲染方法之间的交互效果。

表 15-3　投影类型与渲染方法之间的交互效果

| | 正交投影 | 透视投影 |
|---|---|---|
| z-buffer 渲染法 | CameraViewAngle 属性确定 CameraTarget 属性对应目标点处场景的宽度 | CameraViewAngle 属性确定 CameraPosition 属性对应点到无限远处的场景宽度 |
| 画家渲染法 | 所有对象显示时不考虑 CameraPosition 属性 | 如果图形对象在相机后面，不推荐使用这种投影方式 |

使用正交投影和 z-buffer 渲染器时，相机前面的任何东西都是可见的。使用透视投影时，看到的范围是相机视角对应圆锥体内的范围。

对于在三维空间中移动相机的情况，画家算法不如 z-buffer 渲染法合适，因为 MATLAB 不会沿视轴进行裁剪。用画家算法进行正交投影会导致不管相机在什么位置，场景中的所有对象都可见的现象产生。

### 15.4.3　坐标轴方向上的显示比率

绘图时，MATLAB 会自动根据绘图数据的值和大小确定坐标轴方向上的显示比率，然后绘图并使图形充满可以显示的空间。下面介绍 MATLAB 的默认设置和进行自定义的技巧。

**1．默认设置**

默认时，MATLAB 所创建的用于绘图的坐标系的大小会自动适应图形窗口的大小（但是要略小一点，以留出图边）。如果改变图形窗口的大小，图形坐标系会按比例在各个坐标轴方向上缩放图形，这使得图形总是充满图形窗口中的可用空间。MATLAB 还会设置 $x$，$y$ 和 $z$ 轴方向上的限制来提供各方向上的最大显示宽度。

这个拉伸图形并使之充满绘图空间的要求通常是必要的，但是，有时候也希望能对这个处理过程进行控制，以生成自己想要的结果。例如，不管图形窗口在各个方向上的显示比率是多少，图形都需要按照正确的比例进行显示，或者希望图形在打印页上的大小是固定的。

**2．指定坐标比例**

使用 axis 命令可以调节图形的显示比例。默认时，MATLAB 会查找绘图数据中的最大值和最小值，并选择合适的坐标轴范围。可以通过设置坐标轴方向上的显示范围来替换默认设置，即

```
axis([xmin xmax ymin ymax zmin zmax])
```

可以控制 MATLAB 如何用预定义的 axis 选项比例化坐标轴方向上的数据。

- axis auto　将坐标系各方向上的比例化作默认设置。v=axis 语句将当前图中坐标轴方向上的比例大小保存在向量 $v$ 中。如果希望后面的绘图命令使用与此相同的坐标轴限制，则在命令后面跟上语句 axis(v)。
- axis manual　将比例固定为当前设置。如果随后设置 hold on，则后面的绘图操作会使用当前的比例设置。
- axis tight　根据绘图数据的最大和最小范围设置坐标轴方向上的限制。
- axis ij　将 MATLAB 设成"矩阵"坐标系模式。坐标系统的原点在左上角。$i$ 坐标轴是垂直的，向下为正；$j$ 坐标轴水平，向右为正。
- axis xy　将 MATLAB 设成默认的笛卡儿坐标系模式。坐标系统的原点在左下角。$x$ 轴水平，向右为正；$y$ 轴垂直，向上为正。

**3．指定坐标轴方向上的显示比率**

使用 axis 命令可以调整图形在各坐标轴方向上的显示比率。该命令提供了多个选项来调整图形的显示比率。

- axis equal　坐标系中 $x, y$ 和 $z$ 轴方向上的度量单位相同。
- axis square　使每个坐标轴的长度相等，即图形的包围盒是一个立方体。
- axis vis3d　固定坐标轴各方向上的显示比率，使得可以旋转三维对象。
- axis image　使坐标轴各方向上的显示比率与图像的相同。
- axis auto　将 $x, y$ 和 $z$ 轴的限制设置为自动选择模式。
- axis normal　将当前坐标系的包围盒恢复为大小刚好包围全部数据，删除对单位比例化所作的任何限制。该设置取消 axis square 命令的设置效果。与 axis auto 一起使用，可以

取消 axis equal 命令的设置效果。

axis 命令通过操作坐标系图形对象属性来进行工作。

**4．示例—— axis 命令选项**

下面 3 个图演示了对一个柱面应用 3 个不同的 axis 选项以后的显示效果。创建柱面的语句为

```
t = 0:pi/6:4*pi;
[x,y,z] = cylinder(4+cos(t),30);
surf(x,y,z)
```

axis normal 是默认设置。MATLAB 会自动在各个方向上拉伸图形，并使之充满图形窗口。其效果如图 15-27 所示。

使用 axis square 命令时，不管图形窗口是什么形状，坐标系的 3 个轴长度都相等。其效果如图 15-28 所示。

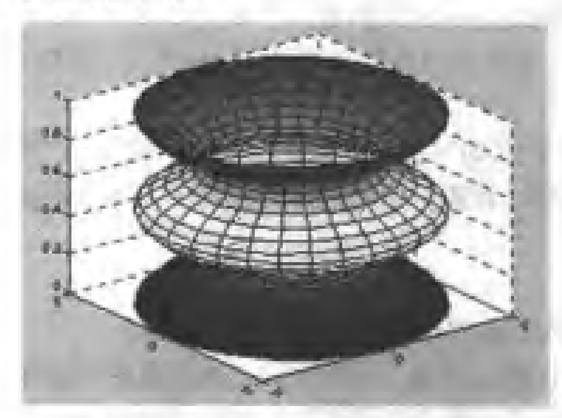


图 15-27　axis normal 命令的设置效果

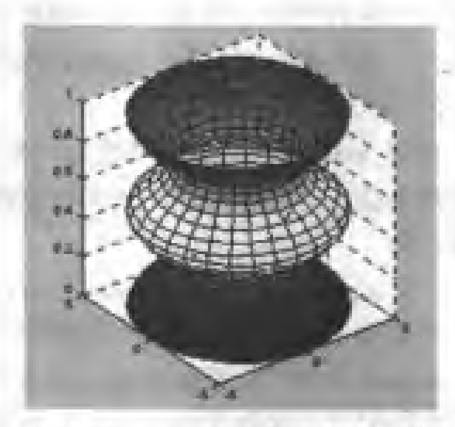


图 15-28　axis square 命令的设置效果

使用 axis equal 命令使图形在各个方向上的度量单位相同。图形效果如图 15-29 所示。

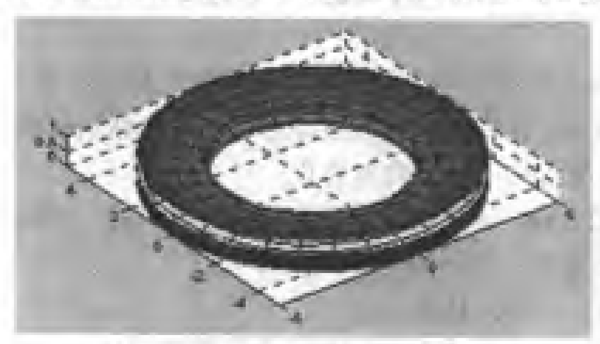


图 15-29　axis equal 命令的设置效果

**5．设置坐标轴方向上显示比率的其他命令**

控制图形在各坐标轴方向上的显示比率有 3 种方法：

- 指定 x, y 和 z 轴方向上的显示比率；
- 指定图形包围盒的形状；
- 指定各坐标轴的显示范围限制。

使用表 15-4 中的命令可以设置这些值。

表 15-4　设置坐标轴方向上显示比率的其他命令

| 命　令 | 功　能 |
|---|---|
| daspect | 设置或查询数据在各坐标轴方向上的显示比率 |
| pbaspect | 设置或查询图形包围盒在各方向上的显示比率 |
| xlim | 设置或查询 x 轴方向上的限制 |
| ylim | 设置或查询 y 轴方向上的限制 |
| zlim | 设置或查询 z 轴方向上的限制 |

### 6. 显示比率属性

axis 命令通过设置不同的坐标系对象属性来进行工作。可以直接设置这些属性，以得到想要的精确效果。表 15-5 中列出了这些属性及它们的功能。

表 15-5　显示比率的属性及其功能

| 属　性 | 功　能 |
|---|---|
| DataAspectRatio | 设置单个坐标轴数据值的相对比例。将 DataAspectRatio 属性值设置为[1 1 1]，以显示对应比例的真实对象。给 DataAspectRatio 属性设置一个值，取消自动图形的自动拉伸功能 |
| DataAspectRatioMode | 使用自动模式时，MATLAB 会选择坐标轴对应的显示比例，使得图形充满可用空间 |
| PlotBoxAspectRatio | 设置坐标系包围盒各方向上的比例。给 PlotBoxAspectRatio 函数指定一个值，取消图形的自动拉伸 |
| PlotBoxAspectRatioMode | 使用自动模式时，MATLAB 将 PlotBoxAspectRatio 属性设置为[1 1 1]，除非另外设置 DataAspectRatio 属性和/或坐标轴方向上的范围限制 |
| Position | 用 1 个四维向量定义坐标系的位置和大小 |
| XLim, YLim, ZLim | 设置各坐标轴数据范围的最小和最大限制 |
| XLimMode, YLimMode, ZLimMode | 使用自动模式时，MATLAB 会选择坐标轴范围限制 |

默认时，MATLAB 会自动确定所有这些属性的值，然后应用自动拉伸填充功能。可以通过给属性指定值或将模式设置为 manual 来替换任何属性的自动设置值。为特定属性选择的值的大小主要与要显示的数据的类型有关。

用 MATLAB 进行可视化的大部分数据是显示为直线或网格的数值型数据或表示真实对象的数据。如果是第 1 种数据，需要在每个坐标轴方向上进行拉伸并填充可以利用的空间。但表示真实对象的数据不管从什么角度观察，都需要有精确的比例表示。

# 第 16 章　MATLAB 提供的科学计算可视化工具

MATLAB 提供了丰富的科学计算可视化函数，如表 16-1 所示。利用这些函数，可以绘制切片图、切片等值线图、流线图、流带图、流沙图、流管图、流锥图、卷曲图、等值面图和等帽盖图等。

表 16-1　MATLAB 提供的科学计算可视化函数

| 函　数 | 功　能 |
| --- | --- |
| coneplot | 绘流锥图 |
| contourslice | 在体切片面板上绘等值线 |
| curl | 计算向量场的卷曲和角速率 |
| divergence | 计算向量场的分支 |
| flow | 生成标量体数据 |
| interpstreamspeed | 根据向量场的大小插入流线顶点 |
| isocaps | 计算等值面的等帽盖 |
| isocolors | 计算等值面顶点的颜色 |
| isonormals | 计算等值面顶点的法线 |
| isosurface | 从体数据中提取等值面 |
| reducepatch | 减少面片小面的个数 |
| reducevolume | 减少体数据集中元素的个数 |
| shrinkfaces | 减小面片小面的大小 |
| slice | 在体中绘制切片面板 |
| smooth3 | 平滑三维数据 |
| stream2 | 计算二维流线数据 |
| stream3 | 计算三维流线数据 |
| streamline | 根据二维或三维向量数据绘制流线图 |
| streamparticles | 根据向量体数据绘制流沙图 |
| streamribbon | 根据向量体数据绘制流带图 |
| streamslice | 根据向量体数据绘制流线图 |
| streamtube | 根据向量体数据绘制流锥图 |
| surf2patch | 将表面数据转换为面片数据 |
| subvolume | 提取体数据的子集 |
| volumebounds | 返回体的坐标和颜色范围限制 |

## 16.1　剖面图

剖面图显示穿过三维体积内的一个截面，这个截面可以是二维横剖面、纵剖面、任意

方向的剖面，也可以是一个曲面。剖面图在工程中有着广泛的应用。比如，在地质上，如果某个剖面在地质构造、工程选址布线等方面有着重要意义，往往要把这个剖面单独绘出来进行研究。多个连续且有一定间隔的二维剖面也能反映三维总体的信息。

### 16.1.1　slice 函数

用 slice 函数绘剖面图，其调用格式为：

- slice(V,sx,sy,sz)　绘体积 **V** 中向量 **sx**,**sy** 和 **sz** 指定的点上沿 $x$, $y$ 和 $z$ 方向的切片。**V** 是一个 $m \times n \times p$ 的体积数组，包含默认位置 $X = 1{:}n$, $Y = 1{:}m$, $Z = 1{:}p$ 处的数据点。向量 **sx**,**sy** 和 **sz** 中的每个元素定义切片在 $x$, $y$ 和 $z$ 方向上的分量。
- slice(X,Y,Z,V,sx,sy,sz)　绘体积 **V** 的切片图。**X**,**Y** 和 **Z** 为三维数组，指定 **V** 的坐标。**X**、**Y** 和 **Z** 必须必须位于各向同性的正交空间中。
- slice(V,XI,YI,ZI)　为由 **XI**,**YI** 和 **ZI** 定义的切片在体积 **V** 中绘数据切片图。**XI**,**YI** 和 **ZI** 为矩阵，定义一个表面，体积在表面点上计算。**XI**,**YI** 和 **ZI** 必须大小相同。
- slice(X,Y,Z,V,XI,YI,ZI)　沿数组 **XI**,**YI** 和 **ZI** 定义的表面绘穿过体积 *V* 的切片。
- slice(...,'method')　指定内插方法。'method'可以是'linear','cubic'或'nearest'。

  linear　指定为线性插值（默认选项）。

  cubic　指定为三次插值。

  nearest　指定为最小距离插值法。
- h = slice(...)　返回表面图形对象的句柄向量。

在区间 $-2 \leqslant x \leqslant 2$, $-2 \leqslant y \leqslant 2$, $-2 \leqslant z \leqslant 2$ 上绘下面函数的图形：

$$v = x \cdot e^{(-x^2-y^2-z^2)}$$

```
[x,y,z] = meshgrid(-2:.2:2, -2:.25:2, -2:.16:2);
v = x.*exp(-x.^2-y.^2-z.^2);
xslice = [-1.2,.8,2]; yslice = 2; zslice = [-2,0];
slice(x,y,z,v,xslice,yslice,zslice)
colormap hsv
```

生成图 16-1。

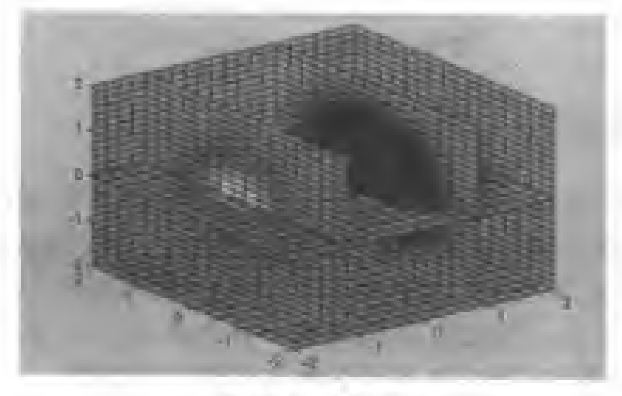
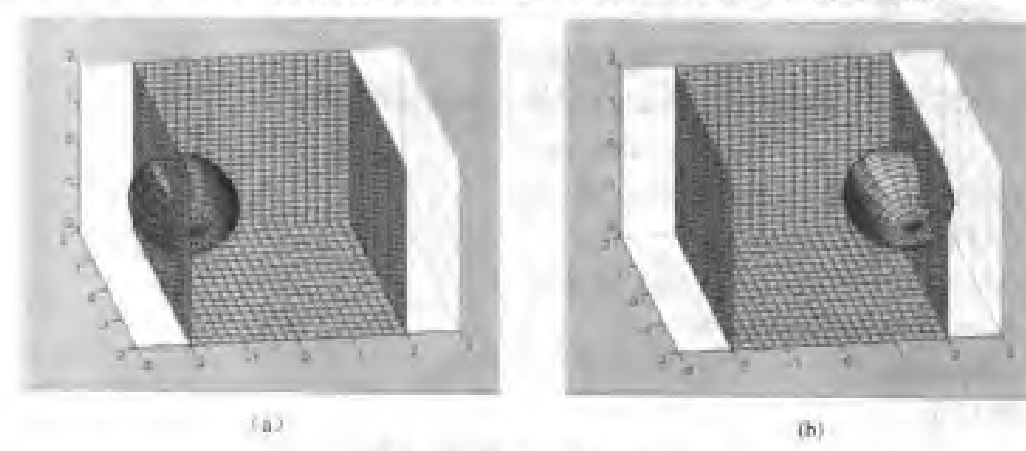


图 16-1　剖面图

也可以用任意形状的表面对体积进行切片。下面的一个例子演示了一个球形切片表面穿过体积的情形。

```
[xsp,ysp,zsp] = sphere;
slice(x,y,z,v,[-2,2],2, -2)
for i = -3:.2:3
    hsp = surface(xsp+i,ysp,zsp);
    rotate(hsp,[1 0 0],90)
    xd = get(hsp,'XData');
    yd = get(hsp,'YData');
    zd = get(hsp,'ZData');
    delete(hsp)
    hold on
    hslicer = slice(x,y,z,v,xd,yd,zd);
    axis tight
    xlim([-3,3])
    view(-10,35)
    drawnow
    delete(hslicer)
    hold off
end
```

图 16-2（a）和（b）分别为球形切片穿越体积时位于左侧和右侧时的情形。



（a）　　（b）

图 16-2　球形切片穿越体积

## 16.1.2　切片等值线图

切片等值线图是绘三维体积图的剖面图并在图上叠加对应位置的等值线图。与切片流线图的功能相似，但它体现的是与流线方向垂直的等势线。用 contourslice 函数实现。

用 contourslice 函数在体积切片面板中绘等值线图。调用格式为：

- contourslice(X,Y,Z,V,Sx,Sy,Sz)　在与 $x$ 轴、$y$ 轴和 $z$ 轴平行的面板中，在向量 ***Sx***、***Sy*** 和 ***Sz*** 所表示的点上绘等值线。数组 ***X,Y*** 和 ***Z*** 定义体积 ***V*** 的坐标。每条等值线的颜色由体积 ***V*** 确定。它必须是一个 $m \times n \times p$ 的体积数组。
- contourslice(X,Y,Z,V,Xi,Yi,Zi)　沿数组 ***Xi,Yi,Zi*** 定义的表面绘穿过体积 ***V*** 的等值线。
- contourslice(V,Sx,Sy,Sz)和 contourslice(V,Xi,Yi,Zi)　（忽略 ***X,Y*** 和 ***Z*** 变量）假设[X,Y,Z]

= meshgrid(1:n,1:m,1:p) 其中 [m,n,p]= size(v)。

- contourslice(...,n)　在每个面板中绘 *n* 条等值线，覆盖自动设置的值。
- contourslice(...,cvals)　在每个面板中由向量 ***cvals*** 指定的位置上绘 length(cval)条等值线。
- contourslice(...,[cv cv])　在水平 *cv* 上计算每个面板中的单条等值线。
- contourslice(...,'method')　指定将要使用的内插方法。内插方法可以是 linear,cubic 或 nearest。nearest 是默认方法。
- h = contourslice(...)　返回阴影对象的句柄向量。

本例使用 flow 数据集绘制切片等值线图。本例中，

- 对于 *Sx*，指定一个 length=9 的向量；对于 *Sy*，指定一个空向量，对于 *Sz*，指定一个值为 0 的标量。这将在 *y*-*z* 面板中沿 *x* 方向创建 9 条等值线，在 *x*-*y* 面板中 *z*=0 处创建一条等值线。
- 使用 linspace 函数定义一个值从-8 到 2 的具有 10 个元素的线性间隔的向量，它指定每个间隔所绘的等值线条数。
- 定义视图和投影类型(camva、camproj 和 campos)。
- 设置图像(gcf)和坐标轴 axes(gca)的属性。

```
[x y z v] = flow;
h = contourslice(x,y,z,v,[1:9],[],[0],linspace(-8,2,10));
axis([0,10, -2,2, -2,2]); daspect([1,1,1])
camva(24); camproj perspective;
campos([-3, -15,5])
set(gcf,'Color',[.5,.5,.5],'Renderer','zbuffer')
set(gca,'Color','black','XColor','white', ...
      'YColor','white','ZColor','white')
box on
```

生成图 16-3。

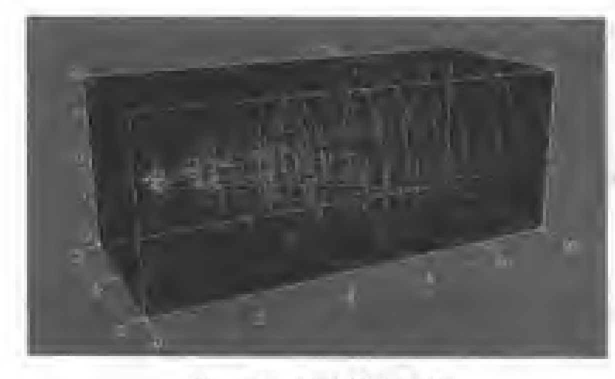


图 16-3　切片等值线图

### 16.1.3　切片流线图

切片流线图是在三维图的图切剖面上叠加流线图。利用该图可以显示三维体积内部某个特定剖面上的流动特征。用 streamslice 函数实现。

用 streamslice 函数在切片面板中绘流线图。调用格式为：

- streamslice(X,Y,Z,U,V,W,startx,starty,startz)　用箭头绘向量数据 ***U,V,W*** 的间隔一定的流线图。数组 ***X,Y,Z*** 定义 ***U,V,W*** 的坐标。***U,V,W*** 必须是 $m \times n \times p$ 的体积数组。

需要注意的是，不能假定流动方向与切片面板平行。如，在常数 *z* 处的流动切片中，当为面板计算流线时，忽略 *z* 方向上的向量场分量 ***W***。

流动切片对于确定流线、流动锥体和流带的起点很有用。

- streamslice(U,V,W,startx,starty,startz)　假设 ***X,Y*** 和 ***Z*** 由下面的语句确定：

```
[X,Y,Z] = meshgrid(1:n,1:m,1:p)
```

其中，[m,n,p] = size(U)。

- streamslice(X,Y,U,V)　根据向量体积数据 ***U*** 和 ***V*** 绘合理分隔的流线。数组 ***X*** 和 ***Y*** 定义 ***U*** 和 ***V*** 的坐标。

- streamslice(U,V)　假设 ***X***, ***Y*** 和 ***Z*** 由下面的语句定义：

```
[X,Y,Z] = meshgrid(1:n,1:m,1:p)
```

其中，[m,n,p] = size(U)。

- streamslice(...,density)　对流线的自动间隔进行修改。density 参数必须大于 0，其默认值为 1；该值越大，在每个面板上创建的流线越多。

- streamslice(...,'arrowsmode')　确定方向箭头是否存在。arrowmode 可以是：

  arrows　在流线上绘方向箭头（默认选项）。

  noarrows　不绘方向箭头。

- streamslice(...,'method')　指定内插方法，可以是：

  linear　线性插值（默认设置）；

  cubic　三次插值；

  nearest　最近邻插值。

- h = streamslice(...)　返回所创建的直线对象的句柄向量。

- [vertices arrowvertices] = streamslice(...)　返回流线和箭头的两个单元数组。可以将这些值传给任何一个流线。

本例利用 wind 数据创建一个 z=5 处的流动切片图。

```
load wind
daspect([1 1 1])
streamslice(x,y,z,u,v,w,[],[],[5])
axis tight
```

生成图 16-4。

本例使用 streamslice 函数计算流线和方向箭头的顶点数据。streamline 函数将利用该数据绘直线和箭头。

```
load wind
daspect([1 1 1])
[verts averts] = streamslice(u,v,w,10,10,10);
streamline([verts averts])
spd = sqrt(u.^2 + v.^2 + w.^2);
hold on;
slice(spd,10,10,10);
```

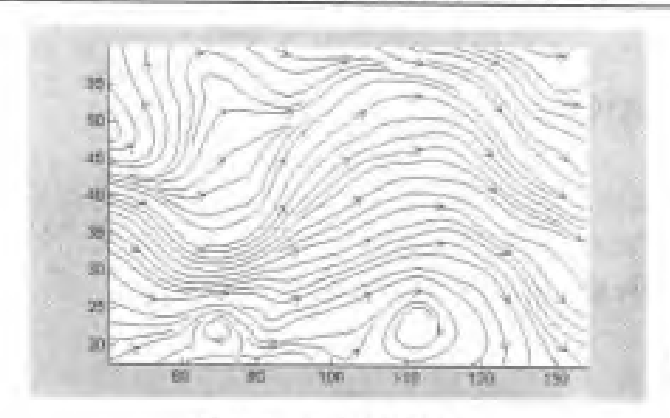


图 16-4　切片流线图之一

```
colormap(hot)
shading interp
view(30,50); axis(volumebounds(spd));
camlight; material([.5 1 0])
```

生成图 16-5。

本例在表面图上添加等值线，然后使用 streamslice 函数绘直线，表示表面的梯度。使用 interp2 函数查找表面上直线的点。

```
z = peaks;
surf(z)
shading interp
hold on
[c ch] = contour3(z,20); set(ch,'edgecolor','b')
[u v] = gradient(z);
h = streamslice(-u, -v);
set(h,'color','k')
for i=1:length(h);
    zi = interp2(z,get(h(i),'xdata'),get(h(i),'ydata'));
    set(h(i),'zdata',zi);
end
view(30,50); axis tight
```

生成图 16-6。

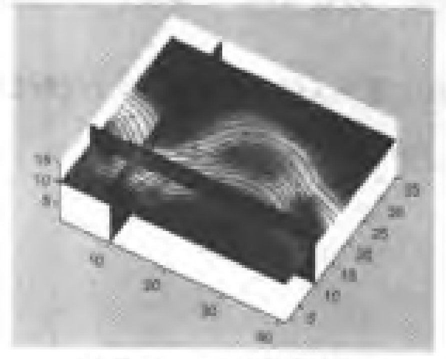


图 16-5　切片流线图之二

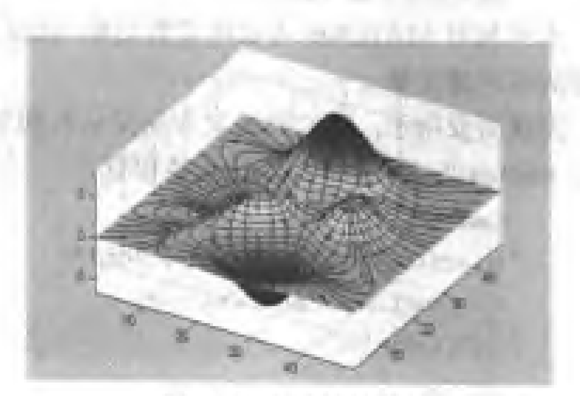


图 16-6　切片流线图之三

## 16.2 表现流动特征

流线图常用于表现流动物体的流动特征。MATLAB 中对流线图有生动的表现。在 MATLAB 中，不仅可以绘制常规的流线图，还可以绘制流锥图、流沙图、流管图、流带图和卷曲图等图形，用多种形式表现物体的流动特征。

### 16.2.1 流线图

用 streamline 函数根据二维和三维向量数据绘流线图。调用格式为：

- h = streamline(X,Y,Z,U,V,W,startx,starty,startz)　绘三维向量数据 ***U*,*V*,*W*** 的流线图。数组 ***X*,*Y*,*Z*** 定义 ***U*,*V*,*W*** 的坐标。startx,starty,startz 定义流线的起点位置。输出变量 ***h*** 包含直线句柄向量，每条流线对应一个句柄。
- h = streamline(U,V,W,startx,starty,startz)　假设数组 ***X*,*Y*** 和 ***Z*** 由[X,Y,Z] = meshgrid (1:N,1:M,1:P)定义，其中[M,N,P] = size(U)。
- h = streamline(XYZ)　假设 ***XYZ*** 为预先计算好的顶点数组的单元数组。
- h = streamline(X,Y,U,V,startx,starty)　绘二维向量数据 ***U*,*V*** 的流线图。数组 ***X*,*Y*** 定义 ***U*,*V*** 的坐标。***startx*** 和 ***starty*** 定义流线的起点位置。输出变量 *h* 包含一个直线句柄向量，一条流线对应一个句柄。
- h = streamline(U,V,startx,starty)　假设数组 ***X*** 和 ***Y*** 由[X,Y] = meshgrid(1:N,1:M)定义，其中 [M,N] = size(U)。
- h = streamline(XY)　假设 ***XY*** 为预先计算好的顶点数组的单元数组。
- streamline(...,options)　指定创建流线时所用的选项。options 可以定义为一个只有一个元素（步长）的向量或有两个元素（步长和流线顶点的最大个数）的向量

[stepsize]

或

[stepsize, max_number_vertices]

如果没有指定值，MATLAB 使用默认值

stepsize = 0.1 (单元的十分之一)

最大顶点个数 = 1000

本例利用 MATLAB 内部所带数据集 wind 绘流线图。装载 wind 数据集，在 MATLAB 主空间中创建变量 *x*, *y*, *z*, *u*, *v* 和 *w*。

流线面板指示，空气的流动方向是从西到东，起点位置在 *x*=80 处(它靠近 *x* 坐标的最小值)。meshgrid 函数生成流线的起点位置。

```
load wind
[sx,sy,sz] = meshgrid(80,20:10:50,0:5:15);
h = streamline(x,y,z,u,v,w,sx,sy,sz);
set(h,'Color','red')
view(3)
```

生成图 16-7。

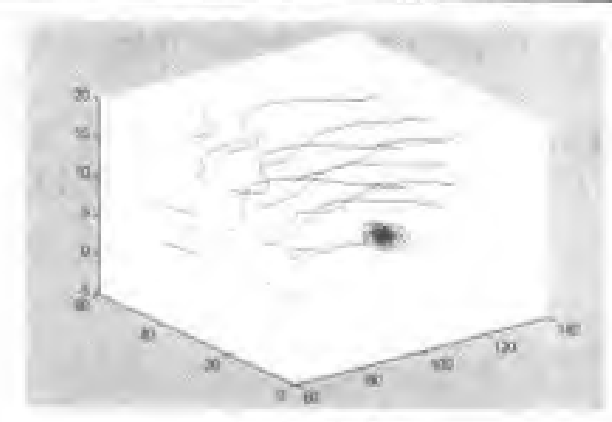


图16-7 流线图

### 16.2.2 流锥图

用coneplot函数在三维向量场中用流锥体绘速率向量图。调用格式为：

- coneplot(X,Y,Z,U,V,W,Cx,Cy,Cz) 用流锥体绘速率向量图。流锥体指向速率向量的方向，其长度与速率向量的大小成比例。

  X,Y,Z 定义向量场的坐标。

  U,V,W 定义向量场。这些数组必须具有相同的大小、单调性和三维特征。

  Cx,Cy,Cz 定义向量场中流锥体的位置。

- coneplot(U,V,W,Cx,Cy,Cz) (忽略 ***X,Y*** 和 **Z** 变量) 假设[X,Y,Z] = meshgrid(1:n,1:m,1:p)，其中，[m,n,p]= size(U)。
- coneplot(...,s) MATLAB自动确定流锥体的显示比例 *s*，使它们能拟合图形的大小。如果没有指定 *s* 的大小，MATLAB将它设为1。令 *s*=0，则不自动设置流锥体的比例。
- coneplot(...,color) 在向量场中进行 **color** 数组插值，并根据插值的值来确定流锥体的颜色。color的大小必须与 ***U,V,W*** 数组的大小相同。该选项只与流锥体一起使用。
- coneplot(...,'quiver') 用箭头代替流锥体绘图。
- coneplot(...,'method') 指定插值方法。可选方法包括：线性插值、二次插值、三次插值和最小距离插值等。默认时选用线性插值。
- coneplot(X,Y,Z,U,V,W,'nointerp') 不将流锥的位置插值到三维区域中，流锥在由 ***X,Y*** 和 **Z** 确定的位置绘制，其方位由 ***U,V,W*** 确定。数组 ***X,Y,Z,U,V*** 和 ***W*** 必须大小相同。
- h = coneplot(...) 返回绘流锥体的阴影对象的句柄。可以使用set命令改变流锥体的属性。

注意：coneplot函数自动设置图中流锥体的显示比例，并使它们与速率向量的大小成比例。调用coneplot函数之前，可以用daspect命令设置数据方向比。

```
daspect([1,1,1])
```

本例演示向量空间数据的速率向量流锥图，该数据表示方形空间区域中气流的流动特征。最后生成的图形采用了一些表现手法，使得数据的显示效果更好，包括：

- 用流锥表示气流速率的大小和方向；
- 数据边界处的分片面板显示了三维区域内部流锥体的承接关系；
- 一定方向的光照提供了不同方位流锥体的视图序列；

- 通过选择视点、投影类型和放大操作来最好地重现数据的内容。

1．装载数据

本例用到 winds 数据集，其中包括 6 个三维数组：*u*, *v* 和 *w* 指定每个点上向量元素，*x*, *y* 和 *z* 指定坐标位置。为了指定流锥体的放置位置，可以首先确定数据的范围。

```
load wind
xmin = min(x(:));
xmax = max(x(:));
ymin = min(y(:));
ymax = max(y(:));
zmin = min(z(:));
```

2．创建流锥体图

确定在数据空间的什么位置绘流锥图。本例分 8 步选择 *x* 和 *y* 的完整范围，在 *z* 中分 4 步选择 3 到 15 的范围。

调用 coneplot 函数之前，使用 daspect 设置坐标轴的数据方向比率，使得 MATLAB 可以确定流锥体的合适大小。

绘流锥体，设置比例因子为 5，使得流锥体比默认大小更大。

设置每个流锥体的颜色（表面色，边缘色）。

```
daspect([2,2,1])
xrange = linspace(xmin,xmax,8);
yrange = linspace(ymin,ymax,8);
zrange = 3:4:15;
[cx cy cz] = meshgrid(xrange,yrange,zrange);
hcones = coneplot(x,y,z,u,v,w,cx,cy,cz,5);
set(hcones,'FaceColor','red','EdgeColor','none')
```

生成图 16-8。

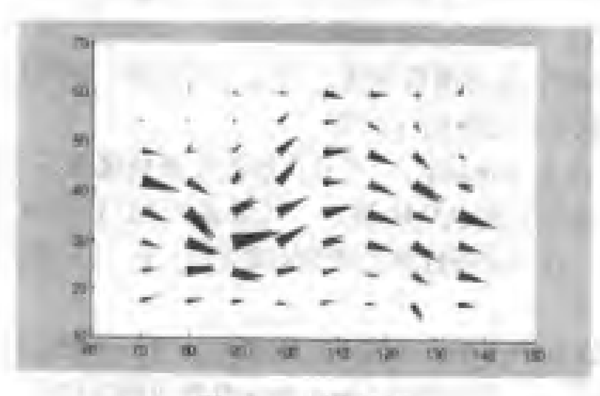


图 16-8　流锥体图

3．添加分片面板

计算向量场的大小（它代表风的速度），为 slice 命令生成标量数据。沿 x 轴在 xmin 和 xmax 处，沿 y 轴在 ymin 和 ymax 处创建分片面板。指定内插面颜色，分片的着色指示风速，不绘边缘。

```
hold on
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hsurfaces = slice(x,y,z,wind_speed,[xmin,xmax],ymax,zmin);
set(hsurfaces,'FaceColor','interp','EdgeColor','none')
hold off
```

生成图 16-9。

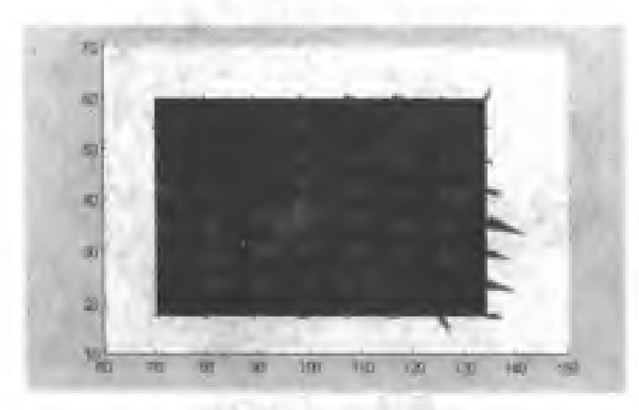


图 16-9　添加切片面板

**4．定义视图**

使用 axis 命令设置坐标轴的范围与数据的范围相等。设置视图方向为 azimuth = 30，elevation = 40，使图尽量大。

```
axis tight; view(30,40); axis off
camproj perspective; camzoom(1.5)
```

生成图 16-10。



图 16-10　定义视图

**5．添加光线**

光源会影响分片面板（表面）和流锥体（阴影），但可以单独设置它们的光照特性。

在相机右侧给光，给流锥和分片面板一个平滑、三维的表面。对于每个分片面板，增加 AmbientStrength 属性的值，改进深蓝色的可视性。

增加 DiffuseStrength 属性的值，加亮显示这些流锥，并显示特殊的反射光影。

```
camlight right; lighting phong
set(hsurfaces,'AmbientStrength',.6)
set(hcones,'DiffuseStrength',.8)
```

生成图 16-11。



图 16-11　添加光线

### 16.2.3　流沙图

MATLAB 可以用流动的点来表示物体的流动，称为流沙图。它用 streamparticles 函数来实现。

用 streamparticles 函数生成和显示流沙图。调用格式为：

- streamparticles(vertices)　绘向量场的流沙图。流动的颗粒通常用标记代表，可以显示流线的位置和速率。**vertices** 为二维或三维向量的单元数组。
- streamparticles(vertices,n)　用 $n$ 确定需要绘多少个流动微粒。ParticleAlignment 属性控制如何解释 $n$。
  - 如果 ParticleAlignment 设置为 off（默认设置），并且 $n$ 大于 1，则在流线上等间距绘近似于 $n$ 个微粒。
  - 如果 $n$ 小于或等于 1，$n$ 解释为原始流动顶点的一部分。例如，如果 $n$=0.2，则近似有 20%的顶点被使用。$n$ 确定所绘顶点个数的上限。注意，微粒的实际个数可能会偏离 $n$。
  - 如果 ParticleAlignment 设置为 on，则 $n$ 确定微粒最多的流线上的微粒个数，并将其他流线上微粒的间隔数设置为此值。默认时，$n$ = 1。
- streamparticles(...,'PropertyName',PropertyValue,...)　用给定的属性和指定值控制流动微粒。任何没有指定的属性按默认值赋值。MATLAB 忽略表 16-2 所示的属性名。

**表 16-2　流动微粒属性表**

| 流动微粒的属性 | 描　述 |
| --- | --- |
| Animate – 流动微粒的运动[非负整数] | 流动微粒的快照次数。默认值为 0，不进行快照。设置为 Inf 时，将一直进行快照，直到按下<ctrl-c>置 |
| FrameRate – 每秒快照帧数[非负数数] | 本属性为快照指定每秒的帧数。设置为 Inf 时，将尽可能快地绘快照图（默认设置）。注意，快照的速度受到计算机速度的限制。此时，FrameRate 属性的值可能不必达到 |
| ParticleAlignment – 将微粒与流线对齐[on\|{off}] | 设置此属性为 on，在每个流线的起点绘微粒。本属性控制 streamparticles 函数如何解释变量 n（流动微粒数） |

流动微粒为直线对象，除了流动微粒属性以外，可以指定任何直线对象属性，如Marker和EraseMode等。streamparticles函数被调用时设置表16-3所示的直线属性。

表16-3 直线属性表

| 直线属性 | Streamparticles函数设定的值 |
|---|---|
| EraseMode | xor |
| LineStyle | none |
| Marker | o |
| MarkerEdgeColor | none |
| MarkerFaceColor | Red |

可以通过指定一个属性名和属性值作为变量来覆盖这些属性值中的任意一个。如下面的语句用RGB值将MarkerFaceColor设置为中灰。

```
streamparticles(vertices,'MarkerFaceColor',[.5 .5 .5])
```

- streamparticles(line_handle,...) 使用由line_handle确认的线性对象绘流沙图。
- h = streamparticles(...) 返回创建的直线对象的句柄向量。

本例将流线和流沙快照组合起来，interpstreamspeed函数控制快照的速度。将坐标轴的DrawMode属性设置为fast，将使快照速度更快。

```
load wind
[sx sy sz] = meshgrid(80,20:1:55,5);
verts = stream3(x,y,z,u,v,w,sx,sy,sz);
sl = streamline(verts);
iverts = interpstreamspeed(x,y,z,u,v,w,verts,.025);
axis tight; view(30,30); daspect([1 1 .125])
camproj perspective; camva(8)
set(gca,'DrawMode','fast')
box on
streamparticles(iverts,35,'animate',10,'ParticleAlignment','on')
```

生成图16-12。

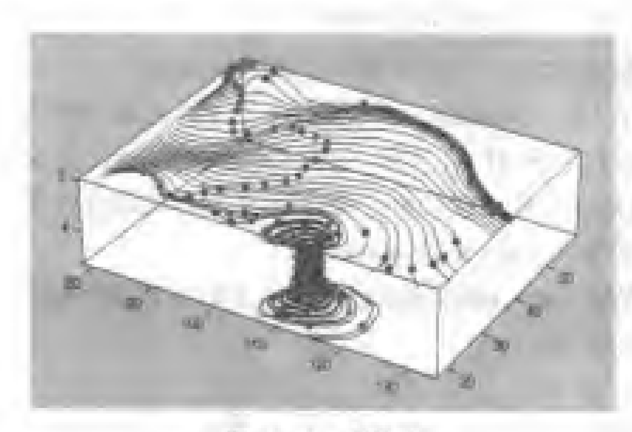


图16-12 流沙图

下面的图片为快照的一张静态视图。本例使用z=5处面板的流线对沿这些直线的流动进行快照。

```
load wind
daspect([1 1 1]); view(2)
[verts averts] = streamslice(x,y,z,u,v,w,[],[],[5]);
sl = streamline([verts averts]);
axis tight off;
set(sl,'Visible','off')
iverts = interpstreamspeed(x,y,z,u,v,w,verts,.05);
set(gca,'DrawMode','fast','Position',[0 0 1 1],'ZLim',[4.9 5.1])
set(gcf,'Color','black')
streamparticles(iverts, 200, ...
    'Animate',100,'FrameRate',40, ...
'MarkerSize',10,'MarkerFaceColor','yellow')
```

生成图 16-13。

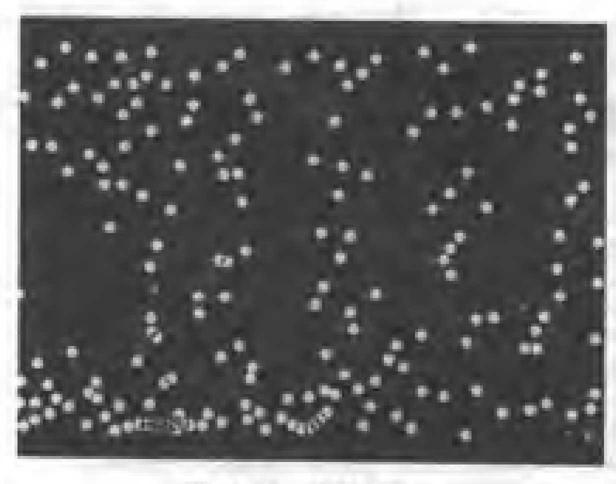


图 16-13　流沙图之二

### 16.2.4　流带图

流带图用飘动的条带来表现物体的流动特征。

用 streamribbon 函数创建一个三维流带图。调用格式为：

- streamribbon(X,Y,Z,U,V,W,startx,starty,startz)　根据向量体积数据 ***U,V*** 和 ***W*** 绘流带图。数组 ***X,Y*** 和 ***Z*** 定义 ***U,V*** 和 ***W*** 的坐标。startx,starty 和 startz 定义条带中心处流带的起点位置。条带的扭曲与向量场的卷曲成比例。条带的宽度自动进行计算。一般地，应该在调用 streamribbon 函数之前设置 DataAspectRatio 属性。
- streamribbon(U,V,W,startx,starty,startz)　假设 ***X,Y*** 和 ***Z*** 由下面的表达式定义。

```
[X,Y,Z] = meshgrid(1:n,1:m,1:p)
```

其中，[m,n,p] = size(U)。

- streamribbon(vertices,X,Y,Z,cav,speed)　假定预先计算的流线顶点、卷曲角速率和流速。vertices 为流动直线顶点的单元数组（就象 stream3 所创建的）。***X,Y,Z,cav*** 和 **speed** 为三维数组。
- streamribbon(vertices,cav,speed)　假定 ***X,Y*** 和 ***Z*** 由下面的语句确定。

```
[X,Y,Z] = meshgrid(1:n,1:m,1:p)
```

其中 [m,n,p] = size(cav)

- streamribbon(vertices,twistangle)　使用向量单元数组 twistangle 确定条带的扭曲，vertices 每个元素的大小和必须和 twistangle 的相同。
- streamribbon(...,width)　设置条带的宽度为 width。
- h = streamribbon(...)　返回句柄向量到表面对象。

本例使用 wind 数据集来绘流带图。输入变量包括坐标、向量场组分和流带的起点位置。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
streamribbon(x,y,z,u,v,w,sx,sy,sz);
%-----定义视图和光照。
axis tight
shading interp;
view(3);
camlight; lighting gouraud
```

生成图 16-14。

### 16.2.5　流管图

流管图则用不同粗细和颜色的空间管道表现物体在空间的流动特征。用 streamtube 函数绘流管图。

用 streamtube 函数创建一个三维流管图。调用格式为：

- streamtube(X,Y,Z,U,V,W,startx,starty,startz)　根据向量体积数据 ***U,V,W*** 绘流管图。数组 ***X,Y,Z*** 定义 ***U,V,W*** 的坐标。startx, starty 和 startz 定义管中心处流线的起点。
- streamtube(U,V,W,startx,starty,startz)　假设 ***X,Y*** 和 **Z** 由下面的语句确定：

```
[X,Y,Z] = meshgrid(1:n,1:m,1:p)
```

其中，[m,n,p] = size(U)。

- streamtube(vertices,X,Y,Z,divergence)　假定预先计算的流线顶点和差异。**vertices** 为流线顶点的单元数组。***X,Y,Z*** 和 **divergence** 为三维数组。
- streamtube(vertices,divergence)　假定 ***X,Y*** 和 **Z** 由下面的语句确定：

```
[X,Y,Z] = meshgrid(1:n,1:m,1:p)
```

其中，[m,n,p] = size(divergence)。

- streamtube(vertices,width)　在向量单元数组 **width** 中指定圆管的宽度。**vertices** 中每一个对应元素的大小必须与 **width** 相等。width 也可以是一个标量，为所有流管的宽度指定大小。
- streamtube(vertices)　自动选择宽度。
- streamtube(...,[scale n])　按 scale 参数确定管体的宽度。默认时 scale = 1。*n* 为沿圆管周边分布的点数。默认时 *n* = 20。

- h = streamtube(...z)　返回一个句柄向量到表面对象。

本例利用 wind 数据集生成流管图。输入包括坐标、向量场分量和流管的起点位置。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
streamtube(x,y,z,u,v,w,sx,sy,sz);
%-----定义视图和光照。
view(3)
axis tight
shading interp;
camlight; lighting gouraud
```

生成图 16-15。

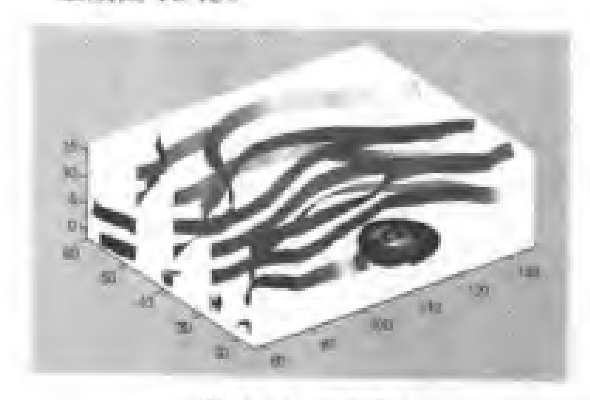


图 16-14　流带图

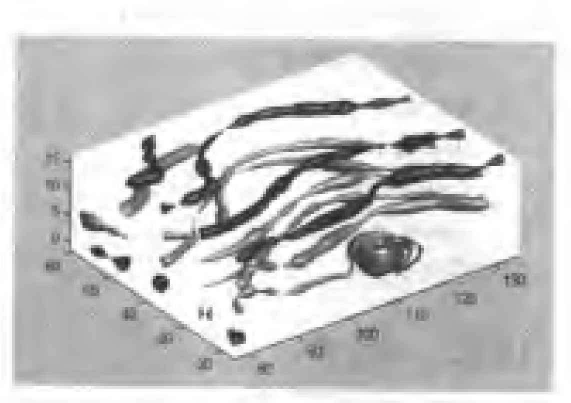


图 16-15　流管图

## 16.2.6　卷曲图

卷曲图通过计算向量场的卷曲和角速率来体现物体的流动特征。用 curl 函数绘图。

用 curl 函数计算向量场的卷曲和角速率。调用格式为：

- [curlx,curly,curlz,cav] = curl(X,Y,Z,U,V,W)　计算垂直于三维向量 ***U,V,W*** 所代表的流场的卷曲和角速率。数组 ***X,Y,Z*** 定义 ***U,V,W*** 的坐标。
- [curlx,curly,curlz,cav] = curl(U,V,W)　假设 ***X,Y*** 和 ***Z*** 由下面的语句确定
  ```
  [X Y Z] = meshgrid(1:n,1:m,1:p)
  ```
  其中，[m,n,p] = size(U)。
- [curlz,cav]= curl(X,Y,U,V)　计算卷曲的 *z* 分量和垂直于二维向量场 ***U,V*** 的 *z* 方向的角速率。数组 ***X,Y*** 定义 ***U,V*** 的坐标。
- [curlz,cav]= curl(U,V)　假设 ***X*** 和 ***Y*** 由下面的语句确定：
  ```
  [X Y] = meshgrid(1:n,1:m)
  ```
  其中，[m,n] = size(U)。
- [curlx,curly,curlz] = curl(...), curlx,curly] = curl(...)　只返回卷曲参数。
- cav = curl(...)　只返回卷曲角速率。

本例使用彩色切片面板在向量场中指定的位置上显示卷曲角速率。

```
load wind
cav = curl(x,y,z,u,v,w);
slice(x,y,z,cav,[90 134],[59],[0]);
shading interp
daspect([1 1 1]); axis tight
colormap hot(16)
camlight
```

生成图 16-16。

下面的例子在体积的一个面板中查看卷曲角速率，并在相同的面板中绘速率向量图。

```
load wind
k = 4;
x = x(:,:,k); y = y(:,:,k); u = u(:,:,k); v = v(:,:,k);
cav = curl(x,y,u,v);
pcolor(x,y,cav); shading interp
hold on;
quiver(x,y,u,v,'y')
hold off
colormap copper
```

生成图 16-17。

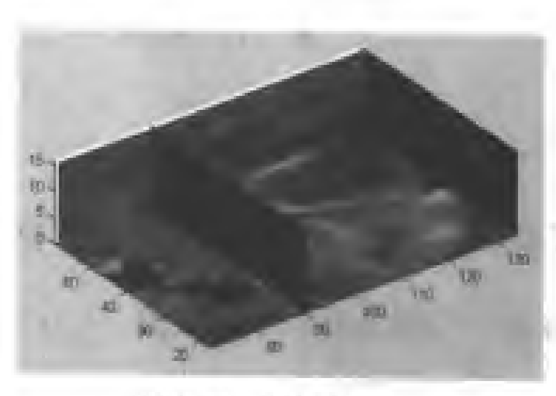


图 16-16　卷曲图之一

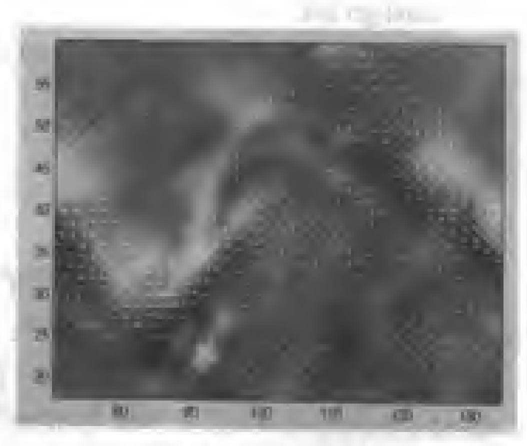


图 16-17　卷曲图之二

## 16.3　等值面

等值面是将体内具有相同值的每个顶点相连以后形成的表面。在强调值相等这一点上，等值面图与等值线图相同。等值面对于确定体内数据的空间分布很有用。

用 isosurface 和 patch 命令创建等值面图。

下面的例子求 M 文件 flow 创建的体中的等值面。用下面的命令生成体数据。

```
[x,y,z,v] = flow;
```

要选择体中相等的值，需要确定体数据中值的范围。

```
min(v(:))
ans =
```

```
    -11.5417
max(v(:))
ans =
    2.4832
```

通过探察，可以选择可以反映数据有用信息的相等值。选定以后，用这个相等的值创建等值面：

- 用 isosurface 命令生成可以直接传递给 patch 函数的数据。
- 重新用体数据的梯度计算表面法向，以获得更好的光照效果。
- 将面片的 FaceColor 属性设置为 red，EdgeColor 属性设置为 none，以生成一个平滑的光照表面。
- 调整视图，添加 phong 光照。

命令行如下所示：

```
hpatch = patch(isosurface(x,y,z,v,0));
isonormals(x,y,z,v,hpatch)
set(hpatch,'FaceColor','red','EdgeColor','none')
daspect([1,4,4])
view([-65,20])
axis tight
camlight left;
set(gcf,'Renderer','zbuffer'); lighting phong
```

得到如图 16-18 所示的等值面。



图 16-18　等值面

## 16.4　等帽盖

三维图形在一定的空间内总是受限的，到了边界处，等值面也就不再往前延伸，而等帽盖用一个平面来表示边界处的这种限制。

下面两个图形显示了等帽盖的使用。图 16-19 是一个没有等帽盖的等值面图。

图 16-20 是图 16-19 添加等帽盖以后的效果。

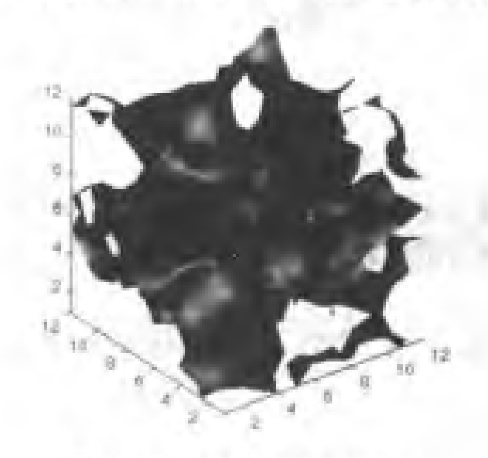


图 16-19　没有等帽盖的等值面

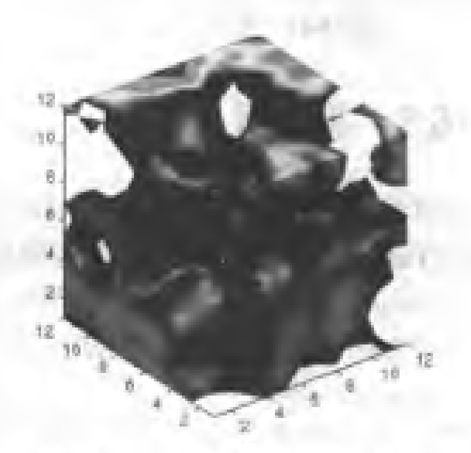


图 16-20　添加等帽盖后的等值面

下面介绍等帽盖的生成过程。就像等值面一样，等帽盖是作为面片对象创建的。使用 isocaps 命令生成数据，然后传递给 patch 函数。例如下面的代码创建值 isoval 处体数据 voldata 的等帽盖。

```
patch(isocaps(voldata,isoval),...
    'FaceColor','interp',...
    'EdgeColor','none')
```

应该用同一套体数据和同一个值创建等值面，以保证等帽盖的边缘与等值面相吻合。

下面的例子演示使用等帽盖时如何设置颜色和光照。有 5 个基本步骤：

- 生成和处理体数据。
- 创建等值面，并设置面片对象的属性以控制颜色和光照。
- 创建等帽盖，设置属性。
- 指定视图。
- 给场景添加光照。

**1．准备数据**

下面用一个随机数组成的三维数组定义体数据，然后对数据进行平滑。

```
data = rand(12,12,12);
data = smooth3(data,'box',5);
```

**2．创建等值面，设置属性**

用 isosurface 和 patch 命令创建等值面，设置着色和光照属性。减少反射光的 AmbientStrength、SpecularStrength 和 DiffuseStrength 表示的强度值，以补偿两个用于提供更多环境光的光源的亮度。

重新计算等值面的顶点法向，以便生成更平滑的光照效果。

```
isoval = .5;
h = patch(isosurface(data,isoval),...
    'FaceColor','blue',...
    'EdgeColor','none',...
    'AmbientStrength',.2,...
    'SpecularStrength',.7,...
    'DiffuseStrength',.4);
isonormals(data,h)
```

**3．创建等帽盖，设置属性**

用生成等值面时使用的体数据和相等值定义等帽盖。指定进行插值着色，选择一个与蓝色等值面具有更好对照效果的颜色查找表。

```
patch(isocaps(data,isoval),...
    'FaceColor','interp',...
    'EdgeColor','none')
colormap hsv
```

**4．定义视图**

将数据在三维坐标轴对应的 3 个方向上的显示比率设置为[1,1,1]，使它们以正确的比例进行显示。用 axis tight 命令剔除坐标系中的白色空间，设置三维视图。

```
daspect([1,1,1])
axis tight
view(3)
```

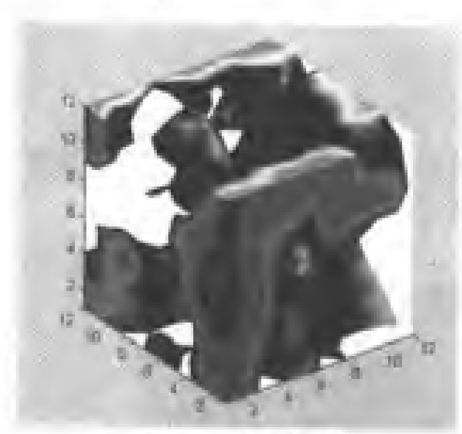


图 16-21　生成等帽盖

5．添加光照

为了在添加比较均匀的光的同时，仍然能显示图形形体的细微变化，本例使用了两个光源，分别在相机的左侧和右侧。使用 phong 光照，使颜色的变化最平滑。phong 光照需要使用 zbuffer 渲染器。

```
camlight right
camlight left
set(gcf,'Renderer','zbuffer');
lighting phong
```

生成的图形如图 16-21 所示。

## 16.5　减少面片上小面的个数

利用 reducepatch 函数可以减少面片小面的个数。该函数的语法格式为：

- reducepatch(p,r)　在保持对象总体形状的条件下减少句柄 *p* 标识的面片的小面个数。MATLAB 根据缩减因子 *r* 的值用两种方法中的一种来解释 *r* 的意义：
  - ➢ 如果 $r<1$，则 *r* 解释为原来小面个数的一部分。例如，如果指定 $r=0.2$，则小面个数减少到原个数的 20%；
  - ➢ 如果 $r\geq1$，则 *r* 是最后小面的个数。例如，如果 $r=400$，则最后小面的个数为 400。
- nfv=reducepatch(p,r)　返回缩减的小面和顶点集，但是不设置面片 p 的 Faces 和 Vertices 属性。结构 nfv 包含缩减后的小面和顶点。
- nfv=reducepatch(fv,r)　对结构 fv 中的小面和顶点进行缩减。
- nfv=reducepatch(p)或 nfv=reducepatch(fv)　使用缩减值 0.5。
- reducepatch(..., 'fast')　假设顶点是惟一的，不计算共享顶点。
- reducepatch(..., 'verbose')　计算时在命令窗口中打印进度信息。
- nfv=reducepatch(f,v,r)　对 *f* 中的小面和 *v* 中的顶点进行缩减。
- [nf,nv]=reducepatch(...)　返回数组 nf 和 nv 中的小面和顶点。

下面的例子演示将小面个数缩减至 15%时的效果。

```
[x,y,z,v] = flow;
p = patch(isosurface(x,y,z,v, -3));
set(p,'facecolor','w','EdgeColor','b');
daspect([1,1,1])
view(3)
figure;
h = axes;
p2 = copyobj(p,h);
```

```
reducepatch(p2,0.15)
daspect([1,1,1])
view(3)
```

生成图 16-22 和图 16-23，分别为缩减前、后的显示外观。

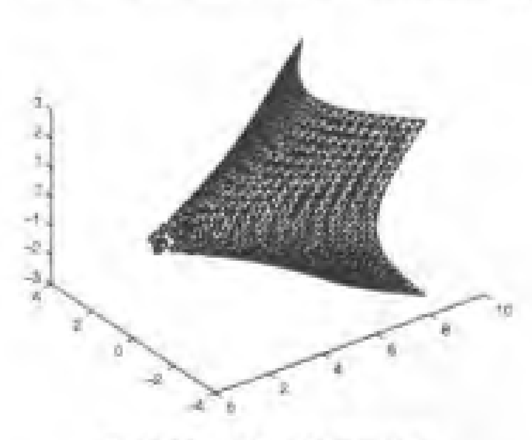


图 16-22　缩减前的显示外观

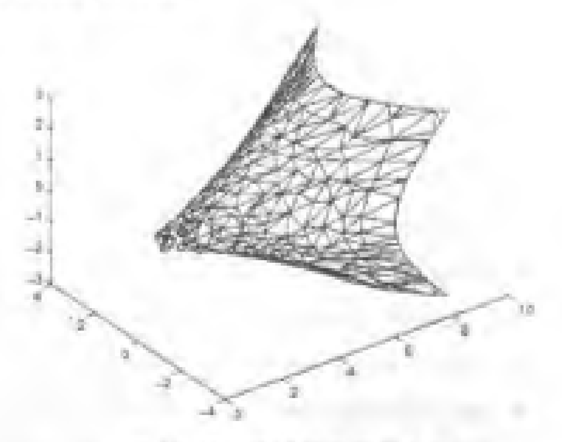


图 16-23　缩减后的显示外观

## 16.6　减少体数据集中元素的个数

用 reducevolume 函数减少体数据集中元素的个数。该函数的语法格式为：

- [nx,ny,nz,nv]=reducevolume(X,Y,Z,V,[Rx,Ry,Rz])　通过保留 $x$ 方向上的第 $Rx$ 个元素，$y$ 方向上的第 $Ry$ 个元素和 $z$ 方向上的第 $Rz$ 个元素来减少体中元素的个数。如果用标量 R 表示元素的个数或减少量，MATLAB 假设减少量为[R R R]。数组 ***X,Y*** 和 **Z** 定义体 ***V*** 的坐标。减少体数据以后的体由变量 *nv* 返回，坐标返回到变量 ***nx, ny*** 和 ***nz*** 中。
- [nx,ny,nz,nv]=reducevolume(V,[Rx,Ry,Rz])　假设数组 ***X, Y*** 和 **Z** 定义为[X,Y,Z]=meshgrid(1:n,1:m,1:p)，其中[m,n,p]=size(V)。
- nv=reducevolume(…)　只返回减少数据后的体。

下面的例子使用了表示人头骨 MRI 切片集的数据集合 mri。

```
load mri
D = squeeze(D);
[x,y,z,D] = reducevolume(D,[4,4,1]);
D = smooth3(D);
p1 = patch(isosurface(x,y,z,D, 5,'verbose'),...
    'FaceColor','red','EdgeColor','none');
isonormals(x,y,z,D,p1);
p2 = patch(isocaps(x,y,z,D, 5),...
  'FaceColor','interp','EdgeColor','none');
view(3); axis tight; daspect([1,1,.4])
colormap(gray(100))
camlight; lighting gouraud
```

生成图 16-24。

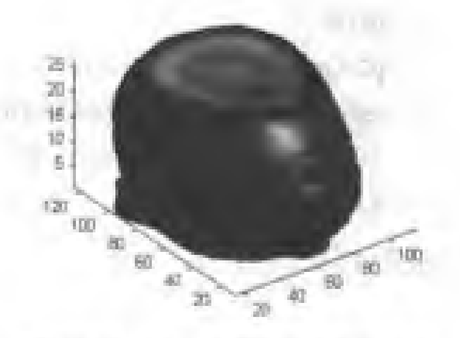


图 16-24　减少体数据集中元素的个数

## 16.7 缩小面片中的小面

用 shrinkfaces 函数缩小面片中的小面。该函数的语法格式为：

- shrinkfaces(p,sf)　缩小面片 p 中小面的面积，缩小的量用缩小因子 *sf* 指定。比如，缩小因子 0.6 指定将小面面积缩小到原来的 60%。如果面片包含公共顶点，则 MATLAB 在缩小小面面积以前会创建非公共顶点。
- nfv=shrinkfaces(p,sf)　将小面和顶点数据返回到结构 nfv 中，但是不设置面片 p 的 Faces 和 Vetices 属性。
- nfv=shrinkfaces(fv,sf)　使用 fv 结构中的小面和顶点数据。
- shrinkfaces(p)和 shrinkfaces(fv)　不指定缩小因子，假设缩小因子为 0.3。
- nfv=shrinkfaces(f,v,sf)　使用数组 *f* 和 *v* 中的小面和顶点数据。
- [nf,nv]=shrinkfaces(...)　将小面和顶点数据返回到两个单独的数组中。

下面的例子使用 flow 数据集。最后生成两个等值面图，分别为缩小小面面积前后的图形。

- 首先，reducevolume 函数进行数据采样，然后，isosurface 函数生成小面和顶点数据。
- patch 函数接受小面/顶点结构 fv 并绘制第一个等值面图。
- 用 daspect,view 和 axis 函数建立视图，然后用 title 函数添加一个标题。
- shrinkfaces 函数修改小面/顶点数据，并将它直接传递给 patch 函数。

在命令窗口键入下面的代码：

```
[x,y,z,v] = flow;
[x,y,z,v] = reducevolume(x,y,z,v,2);
fv = isosurface(x,y,z,v, -3);
p1 = patch(fv);
set(p1,'FaceColor','red','EdgeColor',[.5,.5,.5]);
daspect([1 1 1]); view(3); axis tight
title('Original')

figure
p2 = patch(shrinkfaces(fv,.3));
set(p2,'FaceColor','red','EdgeColor',[.5,.5,.5]);
daspect([1 1 1]); view(3); axis tight
title('After Shrinking')
```

生成图 16-25（a）和（b）。

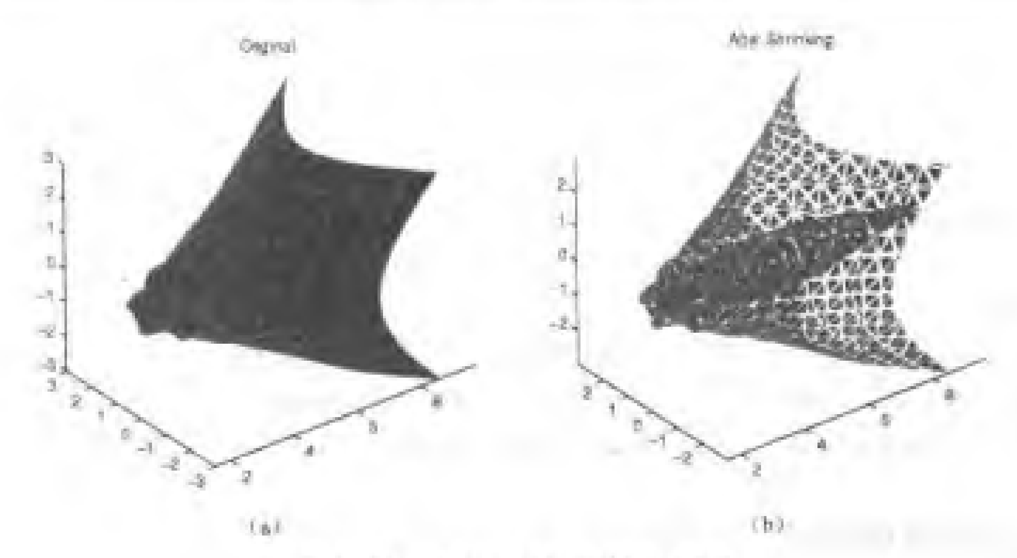


图 16-25 缩小小面前、后的显示外观

## 16.8 子体积

用 subvolume 函数可以提取体数据集的子集。该函数的语法格式为：

- [Nx,Ny,Nz,Nv] = subvolume(X,Y,Z,V,limits) 用 limits 指定的坐标范围提取体数据集 ***V*** 的子集。数组 ***X***, ***Y*** 和 ***Z*** 定义体 ***V*** 的坐标。子体积返回在 NV 中，子体积的坐标由 ***Nx***, ***Ny*** 和 ***Nz*** 给定。
- [Nx,Ny,Nz,Nv] = subvolume(V,limits) 假设数组 ***X***, ***Y*** 和 ***Z*** 由下面的语句定义：

```
[X,Y,Z] = meshgrid(1:N,1:M,1:P)
```

这里，[M,N,P] = size(V)。

- Nv=subvolume(…) 只返回子体积。

下面例子用到的数据取自人头 MRI 切片集。

```
load mri
D = squeeze(D);
[x,y,z,D] = subvolume(D,[60,80,nan,80,nan,nan]);
p1 = patch(isosurface(x,y,z,D, 5),...
      'FaceColor','red','EdgeColor','none');
isonormals(x,y,z,D,p1);
p2 = patch(isocaps(x,y,z,D, 5),...
      'FaceColor','interp','EdgeColor','none');
view(3); axis tight; daspect([1,1,.4])
colormap(gray(100))
camlight right; camlight left; lighting gouraud
```

生成图 16-26。



图 16-26 选取子体积

## 16.9　体包围盒

用 volumebounds 函数返回体的坐标和颜色范围。该函数的语法格式为：

- lims=volumebounds(X,Y,Z,V)　返回当前标量数据坐标系的坐标范围和颜色范围限制。***lims*** 是向量，可表示为[xmin xmax ymin ymax zmin zmax cmin cmax]。可以将这个向量传递给 aixs 命令。
- lims=volumebounds(X,Y,Z,U,V,W)　返回当前向量数据坐标系的坐标范围限制。返回值 ***lims*** 是一个向量，可表示为[xmin xmax ymin ymax zmin zmax]。
- lims=volumebounds(V),lims=volumebounds(U,V,W)　假设 ***X***、***Y*** 和 ***Z*** 由下面的表达式确定：

```
[X Y Z] = meshgrid(1:n,1:m,1:p)
```

这里，[m n p] = size(V)。

下面的例子用 volumebounds 函数设置用 flow 函数生成的等值面的坐标范围和颜色范围。

```
[x y z v] = flow;
p = patch(isosurface(x,y,z,v, -3));
isonormals(x,y,z,v,p)
daspect([1 1 1])
isocolors(x,y,z,flipdim(v,2),p)
shading interp
axis(volumebounds(x,y,z,v))
view(3)
camlight
lighting phong
```

生成图 16-27。

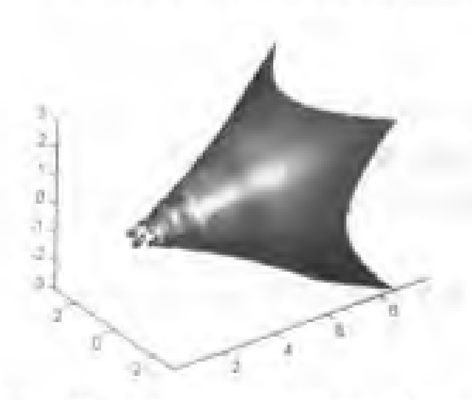


图 16-27　获取体的坐标范围和颜色范围

## 16.10　计算几何问题

MATLAB 还提供了几个用于解决计算几何问题的函数，利用它们，可以求取点集的凸包，进行 Delaunay 剖分和进行 Voronoi 图分析。

### 16.10.1　散点数据的三角化和插值

MATLAB 提供了多个函数来进行最近点问题分析和几何分析，如表 16-4 中所示。

表 16-4　最近点问题分析和几何分析函数

| 函　数 | 描　述 |
|---|---|
| convhull | 凸包 |
| delaunay | Delaunay 三角化 |
| delaunay3 | 3 维 Delaunay 剖分 |
| dsearch | Delaunay 三角化的最近点搜索 |

续表

| 函　　数 | 描　　述 |
|---|---|
| inpolygon | 测试点是否位于多边形内部 |
| polyarea | 多边形区域 |
| rectint | 两个或多个矩形的交 |
| tsearch | 最近三角搜索 |
| voronoi | Voronoi 图 |

### 1. 凸包

convhull 函数返回数据集中点的索引号，用 plot 函数绘 convhull 函数输出数据的图形。

下面的例子载入 seamount 数据，并用散点图绘制经度数据（$x$）和纬度数据（$y$）。然后生成凸包，并用 plot 函数绘制它。

```
load seamount
plot(x,y,'.','markersize',10)
k = convhull(x,y);
hold on, plot(x(k),y(k),'-r'), hold off
grid on
```

生成图 16-28。

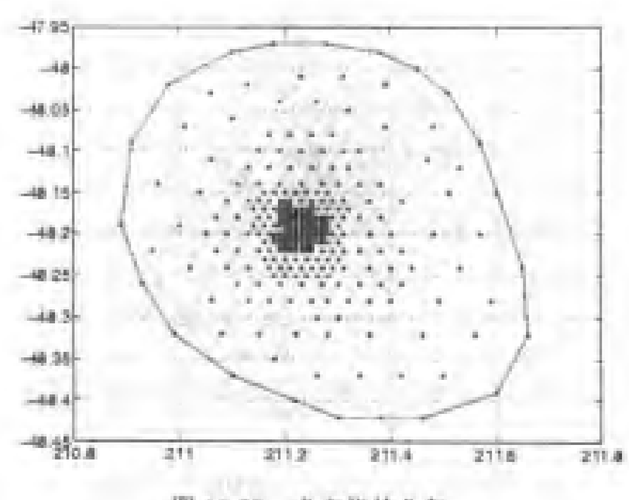



图 16-28　求点集的凸包

### 2. Delaunay 三角化

Delaunay 三角化是目前非常流行的三角化方法，它的最大优点是满足"最大-最小角"优化原则，即使得每个三角形尽可能接近等边三角形。用 delaunay 函数可以返回给定点集的 Delaunay 三角形。下面结合一个实例进行介绍。

（1）绘制 Delaunay 三角形

首先载入 seamount 数据集，并用散点图查看经度（$x$）和纬度（$y$）的数据。

```
load seamount
plot(x,y,'.','markersize',12)
xlabel('Longitude'), ylabel('Latitude')
grid on
```

生成图 16-29。

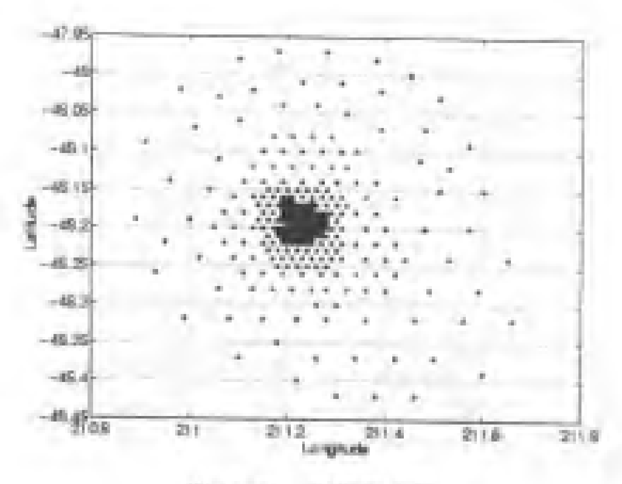



图 16-29　数据散点图

应用 Delaunay 三角化，并用 triplot 函数在散点图上叠加生成的三角形。

```
tri = delaunay(x,y);
hold on, triplot(tri,x,y), hold off
```

如图 16-30 所示。

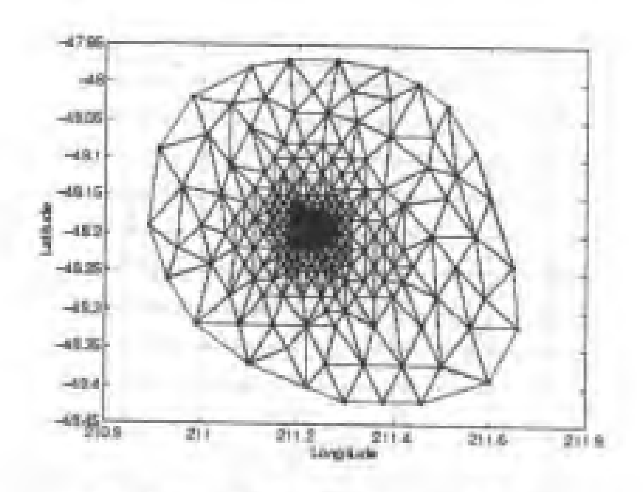



图 16-30　对点集进行 Delaunay 三角化

（2）网格和曲面图

从 seamout 数据集中获取深度数据（z）并添加到 Delaunay 三角形上，用 trimesh 函数生成三维网格。类似地，可以用 trisurf 函数生成曲面。

```
figure
hidden on
trimesh(tri,x,y,z)
grid on
xlabel('Longitude'); ylabel('Latitude'); zlabel('Depth in Feet')
```

生成图 16-31。

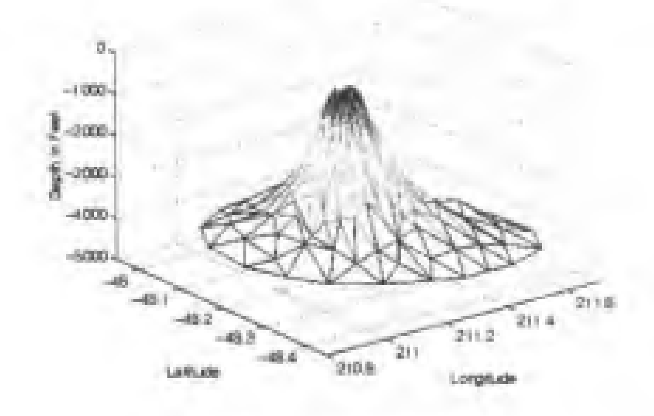



图 16-31 生成三维网格

（3）等值线图

用 meshgrid,griddata 和 contour 函数可以生成 seamount 数据的等值线图，如下所示：

```
figure
[xi,yi] = meshgrid(210.8:.01:211.8, -48.5:.01: -47.9);
zi = griddata(x,y,z,xi,yi,'cubic');
[c,h] = contour(xi,yi,zi,'b-');
clabel(c,h)
xlabel('Longitude'), ylabel('Latitude')
```

生成图 16-32。

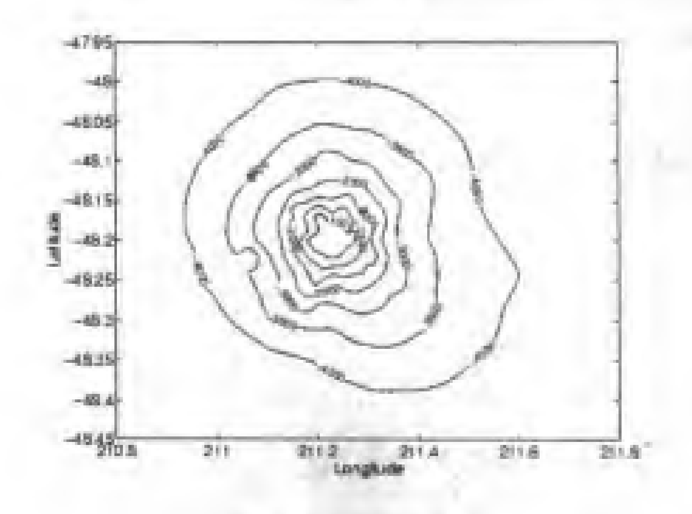



图 16-32 等值线图

（4）最近点搜索

用 dsearch 和 tsearch 函数在 Delaunay 三角形数据中进行搜索：

- dsearch 函数在 Delaunay 三角形数据中查找与指定点最近的点（x, y）的索引号。下面的代码在 seamount 数据集的三角化数据中查找离（211.32, -48.35）最近的点。

```
xi = 211.32; yi = -48.35;
p = dsearch(x,y,tri,xi,yi);
```

```
[x(p), y(p)]
ans =
     211.3400   -48.3700
```

● tsearch 函数查找至 delaunay 函数输出的索引号，这些索引号指定包围给定点的三角形。下面的例子用包围点（211.32，-48.35）的三角形的索引号获取三角形的顶点坐标。

```
xi = 211.32; yi = -48.35;
t = tsearch(x,y,tri,xi,yi);
r = tri(t,:);
A = [x(r) y(r)]
A =
   211.3000   -48.3000
   211.3400   -48.3700
   211.2800   -48.3200
```

### 3. Voronoi 图

Voronoi 图是一种与 Delaunay 三角形有关的最近点绘图技术。对于 coplanar 点集中的每一个点，都可以绘制一个包围中间点的多边形，中间点到该点的距离比到其他任何点的距离都小。这样的多边形称为 Voronoi 多边形，给定点集的所有 Voronoi 多边形构成的集合称为 Voronoi 图。

用 vorvonoi 函数可以绘制 Voronoi 图的单元，或者返回图的边的顶点。下面的例子载入 seamount 数据，然后用 voronoi 函数生成经度数据（$x$）和纬度数据（y）的 Voronoi 图。注意，voronoi 函数只绘制 Voronoi 图的边界单元。

```
load seamount
voronoi(x,y)
grid on
xlabel('Longitude'), ylabel('Latitude')
```

生成图 16-33。

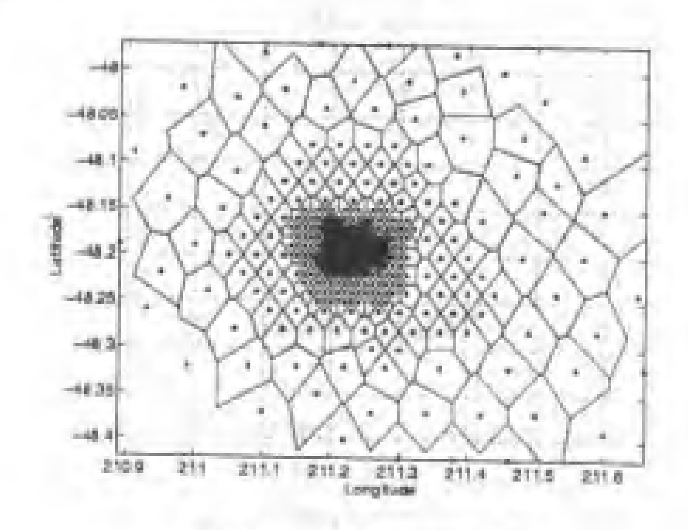



图 16-33　点集的 voronoi 图

## 16.10.2　高维散点集的剖分和插值

科学、工程、统计和算术方面的许多应用都需要用到诸如凸包、Voronoi 图和 Delaunay 剖分之类的结构。MATLAB 提供了多个函数来获取这些结构，如表 16-5 中所示。

表 16-5　高维几何分析函数

| 函　数 | 描　述 |
| --- | --- |
| convhulln | N 维凸包 |
| delaunayn | N 维 Delaunay 剖分 |
| dsearchn | N 维最近点搜索 |
| griddatan | N 维数据网格化和超曲面拟合 |
| tsearchn | N 维最近单纯形搜索 |
| voronoin | N 维 Voronoi 图 |

### 1．凸包

*n* 维空间中数据集的凸包定义为包含数据集的最小顶点区域。convhulln 函数返回组成凸包各小面的数据集中点的索引号。例如，假设 *X* 是一个组成立方体 8 个顶点的 8×3 矩阵，则 *X* 的凸包由 12 个小面组成。

```
d = [-1 1];
[x,y,z] = meshgrid(d,d,d);
X = [x(:),y(:),z(:)];          % 立方体的 8 个顶点
C = convhulln(X)
C =
3    1    5
1    2    5
2    1    3
7    3    5
8    7    5
7    8    3
6    8    5
2    6    5
6    2    8
8    4    3
4    2    3
2    4    8
```

因为数据是三维的，所以组成凸包的小面是三角形的。*C* 的 12 行表示 12 个三角形。*C* 的元素是 *X* 中各点的索引号。例如，第一行 3, 1, 5 表示第一个三角形将 *X*(3,:)、*X*(1,:)和 *X*(5,:)作为它的顶点。

对于三维凸包，可以用 trisurf 函数绘图。但是，用 patch 函数绘图可以对小面的颜色有更多的控制。

下面的代码通过将三角形绘制为三维面片来绘制凸包。

```
figure, hold on
d = [1 2 3 1];          % 至 C 各列的索引
for i = 1:size(C,1)     % 绘制每个三角形
   j= C(i,d);           % 获取 C 的第 i 个元素来构造面片
      h(i)=patch(X(j,1),X(j,2),X(j,3),i,'FaceAlpha',0.9);
end                     % 用'FaceAlpha'属性设置透明性
hold off
view(3), axis equal, axis off
camorbit(90, -5);       % 从另外一个角度视图
title('Convex hull of a cube')
```




图 16-34　立方体的凸包

生成图 16-34。

## 2. Delaunay 剖分

Delaunay 剖分形成一系列单形体，包括二维空间中的三角形和三维空间中的四面体。对于其中的每个单形体，包围该单形体的惟一圆形或球体不包含数据点。

用 delaunayn 函数返回数据集中点的索引号。下面的例子结合立方体的顶点矩阵 ***X*** 进行讨论。

```
d = [-1 1];
[x,y,z] = meshgrid(d,d,d);
X = [x(:),y(:),z(:)];         % 立方体的 8 个顶点
X(9,:) = [0 0 0];             % 给顶点列表中添加中心点
T = delaunayn(X)              % 生成 Delaunay 剖分
T =
     9     1     5     6
     3     9     1     5
     2     9     1     6
     2     3     9     4
     2     3     9     1
     7     9     5     6
     7     3     9     5
     8     7     9     6
     8     2     9     6
     8     2     9     4
     8     3     9     4
     8     7     3     9
```

***T*** 的 12 行表示 12 个单形体，对于不规则四面体的情况，它可以分离立方体。每一行表示一个四面体，并且行的元素为 ***X*** 中各点的索引号。

对于三维剖分，可以用 tetramesh 函数绘图。但是，用 patch 函数绘图对于小面的颜色可以有更多的控制。

下面的代码通过用三维面片绘四面体来绘制剖分结果 ***T***。

```
figure, hold on
d = [1 1 1 2; 2 2 3 3; 3 4 4 4];     % 至 T 的索引
for i = 1:size(T,1)          % 绘制每个 tetrahedron.
   y = T(i,d);               % 获取第 i 个 T 元素，构造面片
   x1 = reshape(X(y,1),3,4);
   x2 = reshape(X(y,2),3,4);
   x3 = reshape(X(y,3),3,4);
   h(i)=patch(x1,x2,x3,(1:4)*i,'FaceAlpha',0.9);
end
hold off
view(3), axis equal
axis off
camorbit(65,120)             % 从另一个角度视图
title('Delaunay tessellation of a cube with a center point')
```

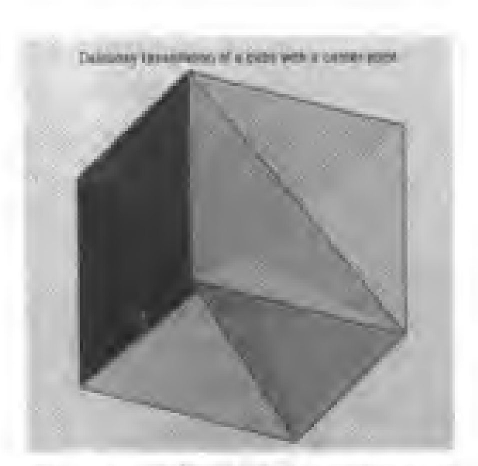


图 16-35　具有一个中心点的立方体的 Delaunay 剖面

生成图 16-35。可以用 cameramenu 函数任意旋转图形。

3．Voronoi 图

在 $n$ 维空间中给定 $m$ 个数据点，Voronoi 图将 $n$ 维空间分成 $m$ 个多边形区域，一个区域对应于一个数据点。这个区域称为 Voronoi 单元。Voronoi 单元满足包含所有到该点比到其他点距离更近的点的条件。

用 voronoin 函数返回一个 $\boldsymbol{V}$ 矩阵和一个 $\boldsymbol{C}$ 向量。$\boldsymbol{V}$ 矩阵的大小为 $m \times n$，表示 $n$ 维空间中的 $m$ 个点。$\boldsymbol{V}$ 的每一行表示一个 Voronoi 顶点。$\boldsymbol{C}$ 向量中的元素为单元数组。单元数组 $\boldsymbol{C}$ 中的每个向量表示一个 Voronoi 单元。该向量包含 $\boldsymbol{V}$ 中所有点的索引号，这些点是 Voronoi 单元的顶点。每一个 Voronoi 单元可能会有不同的点数。

因为 Voronoi 单元可以是没有边界的，$\boldsymbol{V}$ 的第一行是无穷远处的一个点。然后，$\boldsymbol{C}$ 中的任何无边界 Voronoi 单元都包含无穷远处的这个点，即 $\boldsymbol{V}$ 中的第一个点。

下面还是以立方体及其中心点为例进行讨论，添加了随机噪声，使得立方体不很规则。生成的 Voronoi 图有 9 个 Voronoi 单元。

```
d = [-1 1];
[x,y,z] = meshgrid(d,d,d);
X = [x(:),y(:),z(:)];            % 生成立方体的 8 个顶点
X(9,:) = [0 0 0];                % 在顶点列表中添加中心点
X = X+0.01*rand(size(X));        % 使立方体不规则
[V,C] = voronoin(X)
V =
      Inf         Inf         Inf
   0.0055      1.5054      0.0004
   0.0037      0.0101     -1.4990
   0.0052      0.0087     -1.4990
   0.0030      1.5054      0.0030
   0.0072      0.0072      1.4971
```

```
   -1.7912    0.0000    0.0044
   -1.4886    0.0011    0.0036
   -1.4886    0.0002    0.0045
    0.0101    0.0044    1.4971
    1.5115    0.0074    0.0033
    1.5115    0.0081    0.0040
    0.0104   -1.4846   -0.0007
    0.0026   -1.4846    0.0071
C =
    [1x8 double]
    [1x6 double]
    [1x4 double]
    [1x6 double]
    [1x6 double]
    [1x6 double]
    [1x6 double]
    [1x6 double]
    [1x12 double]
```

本例中，***V*** 是一个 13×3 的矩阵，这 13 行是 13 个 Voronoi 顶点的坐标。***V*** 的第一行是无穷远处的点。***C*** 是 9×1 的单元数组，数组中的每个单元包含一个到 ***V*** 的索引向量，对应于 9 个 Voronoi 单元中的一个。例如，Voronoi 图的第 9 个单元为：

```
C{9} = 2 3 4 5 6 7 8 9 10 11 12 13
```

如果单元数组的单元中任何一个的索引号为 1，则对应的 Voronoi 单元包含 ***V*** 中的第一个点，即无穷远处的点。这表示这个 Voronoi 单元是没有边界的。

要查看有界 Voronoi 单元，即不包含无穷远处点的单元，用 convhulln 函数计算组成 Voronoi 单元的小面顶点。然后用 patch 函数和其他绘图函数生成图形。例如，下面的代码绘制由 *C* 中第 9 个单元定义的 Voronoi 单元。

```
X = V(C{9},:);      % 查看第 9 个 Voronoi 单元
K = convhulln(X);
figure
hold on
d = [1 2 3 1];       % 至 K 的索引
for i = 1:size(K,1)
   j = K(i,d);
   h(i)=patch(X(j,1),X(j,2),X(j,3),i,'FaceAlpha',0.9);
end
hold off
view(3)
axis equal
title('One cell of a Voronoi diagram')
```

生成图 16-36。

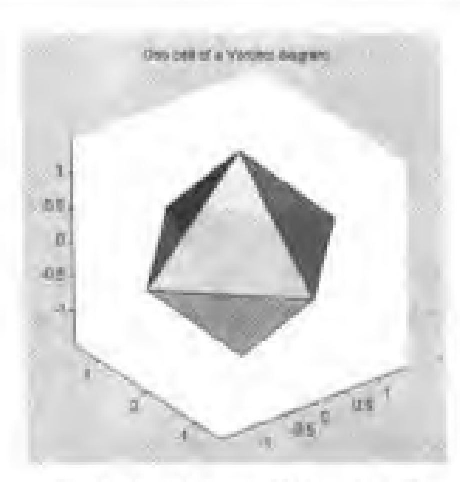



图 16-36　Voronoi 图的一个单元

#### 4. 高维数据的插值

用 griddatan 函数对高维数据，特别是离散数据插值。该函数用 delaunayn 函数剖分数据，然后基于剖分结果进行插值。

下面的例子根据 *n* 个离散点及这些离散点上的函数值来显示函数的图形。***X*** 是离散点的坐标矩阵，大小为 $n\times3$，每一行包含一个点的（x, y, z）坐标值。向量 ***v*** 包含这些点上的 *n* 个函数值。本例的函数是各点到原点的距离函数，即 v = x.^2 + y.^2 + z.^2。

首先，在三维空间中生成 500 个随机点，并计算这些点上的函数值。

```
n = 500;
X = 2*rand(n,3) -1;
v = sum(X.^2,2);
```

下一步通过插值计算网格上的函数值。用 meshgrid 函数创建网格，用 griddatan 函数进行插值。

```
delta = 0.05;
d = -1:delta:1;
[x0,y0,z0] = meshgrid(d,d,d);
X0 = [x0(:), y0(:), z0(:)];
v0 = griddatan(X,v,X0);
v0 = reshape(v0, size(x0));
```

然后，用 isosurface 函数和相关函数绘制等值面，这个等值面上的点具有相同的函数值。例子中用的是 0.6，可以选择其他任何值。因为函数是到原点的距离函数，所以最后生成的曲面是一个球面。

```
p = patch(isosurface(x0,y0,z0,v0,0.6));
isonormals(x0,y0,z0,v0,p);
set(p,'FaceColor','red','EdgeColor','none');
view(3);
camlight;
lighting phong
axis equal
title('Interpolated sphere from scattered data')
```

生成图 16-37。

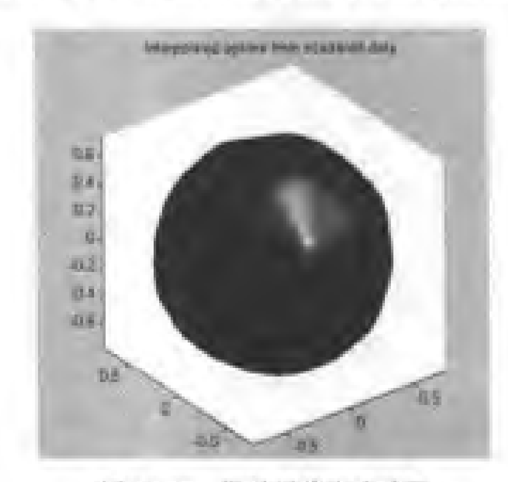


图 16-37　通过插值生成球面